



10



И. Г. Семакин
Е. К. Хеннер
Т. Ю. Шеина

ИНФОРМАТИКА

БАЗОВЫЙ УРОВЕНЬ

И. Г. Семакин, Е. К. Хеннер,
Т. Ю. Шеина

ИНФОРМАТИКА

10 класс

БАЗОВЫЙ УРОВЕНЬ

Учебник

Допущено
Министерством просвещения
Российской Федерации

4-е издание, стереотипное

Москва
«Просвещение»
2022

УДК 373.167.1:004+004(075.3)
ББК 32.81я721
С30

Издание выходит в pdf-формате.

Семакин, Игорь Геннадьевич.
С30 Информатика. 10 класс : базовый уровень : учебник : издание в pdf-формате / И. Г. Семакин, Е. К. Хеннер, Т. Ю. Шеина. — 4-е изд., стер. — Москва : Просвещение, 2022. — 264 с. : ил.

ISBN 978-5-09-101606-2 (электр. изд.). — Текст : электронный.

ISBN 978-5-09-087808-1 (печ. изд.).

Учебник предназначен для изучения курса информатики на базовом уровне в 10 классах общеобразовательных организаций. Содержание учебника опирается на изученный в основной школе (в 7–9 классах) курс информатики. Рассматриваются теоретические основы информатики: понятие информации, информационные процессы, измерение информации, кодирование и обработка информации в компьютере. Излагаются принципы структурного программирования, язык программирования Паскаль. В состав учебника входит практикум, структура которого соответствует содержанию теоретического раздела.

Учебник входит в учебно-методический комплект, включающий учебник для 11 класса, методическое пособие для учителя, электронные материалы.

Соответствует Федеральному государственному образовательному стандарту среднего общего образования и Примерной основной образовательной программе среднего общего образования.

УДК 373.167.1:004+004(075.3)
ББК 32.81я721

Учебное издание

Семакин Игорь Геннадьевич
Хеннер Евгений Карлович
Шеина Татьяна Юрьевна

ИНФОРМАТИКА

10 класс

Базовый уровень

Учебник

Центр математики

Ответственный за выпуск *Е. В. Баклашова*. Ведущий редактор *О. А. Полежаева*

Редактор *Е. В. Баклашова*. Художественный редактор *Н. А. Новак*

Иллюстрации *Я. В. Соловцовой*. Компьютерная вёрстка *В. А. Носенко*

Технический редактор *Е. В. Денюкова*. Корректор *Е. Н. Клитина*.

В оформлении учебника использовано изображение, приведённое на сайте m24.ru (с. 81).

Подписано в печать 11.08.2021. Формат 70×100/16. Усл. печ. л. 21,45.

Тираж экз. Заказ № .

Акционерное общество «Издательство «Просвещение». Российская Федерация, 127473, г. Москва, ул. Краснопролетарская, д. 16, стр. 3, этаж 4, помещение I.

Адрес электронной почты «Горячей линии» — vopros@prosv.ru.

ISBN 978-5-09-101606-2 (электр. изд.) © АО «Издательство «Просвещение», 2020
ISBN 978-5-09-087808-1 (печ. изд.) © Художественное оформление.
АО «Издательство «Просвещение», 2020
Все права защищены

Введение

Что изучается в курсе информатики для 10–11 классов

Изучение любого школьного предмета можно сравнить со строительством дома. Только этот дом складывается не из кирпичей и бетонных плит, а из знаний и умений. Строительство дома начинается с фундамента. Очень важно, чтобы фундамент был прочным, потому что на него опирается всё остальное сооружение. Фундаментом для курса «Информатика 10–11» являются знания и умения, которые вы получили, изучая **информатику в основной школе в 7–9 классах**. Вам уже не требуется объяснять, что такое компьютер и как он работает, с какой информацией может работать компьютер, что такое программа и программное обеспечение компьютера, что такое информационные технологии. В курсе информатики основной школы вы получили представление о том, в каком виде хранится информация в памяти компьютера, что такое алгоритм, информационная модель. Вы научились обращаться с клавиатурой, мышью, дисками, принтером, работать в среде операционной системы, получили основные навыки работы с текстовыми и графическими редакторами, с базами данных и электронными таблицами. Все эти знания и навыки вам будут необходимы при изучении курса «Информатика 10–11».

Термин «информатика» может употребляться в двух смыслах:

- информатика как научная область, предметом изучения которой являются информация и информационные процессы, в которой осуществляется изобретение и создание новых средств работы с информацией;
- информатика как практическая область деятельности людей, связанная с применением компьютеров для работы с информацией.

Как современная техника немыслима без открытий теоретической физики, так и развитие информатики и информационных технологий невозможно без теории информации, теории алгоритмов и целого ряда других теорий в области кибернетики, лингвистики, семиотики, системологии и прочих наук.

В соответствии с современным пониманием в информатике можно выделить четыре части:

- 1) теоретическая информатика;
- 2) средства информатизации;
- 3) информационные технологии;
- 4) социальная информатика.

Теоретическая информатика — это научная область, предмет изучения которой — информация и информационные процессы. Как любая фундаментальная наука, теоретическая информатика раскрывает законы и принципы в своей предметной области.

Вторую и третью части в совокупности можно назвать прикладной информатикой. **Прикладная информатика** — это область практического применения понятий, законов и принципов, выработанных теоретической информатикой. Прикладная информатика, безусловно, связана с применением компьютеров и информационных технологий. В наше время таких прикладных областей очень много: это решение научных задач с помощью компьютера, издательская деятельность, разработка информационных систем, управление различными объектами и системами, техническое проектирование, компьютерное обучение, сетевые информационные технологии и многое-многое другое.

В последние годы в информатике сформировалось новое направление, которое называют **социальной информатикой**. Его появление связано с тем, что широкое внедрение в жизнь компьютерных технологий и современных средств информационных коммуникаций (Интернета, сотовой связи) оказывает всё более сильное влияние на общество в целом и на каждого отдельного человека. Общественное развитие движется к своей новой ступени — к *информационному обществу*.

Предметная область современной информатики очень велика и разнообразна. Как известно, нельзя объять необъятное. И наш курс затронет лишь часть тем и задач информатики. Вопросы,

которые мы с вами будем изучать, относятся к четырем важнейшим понятиям информатики:

- 1) информационные процессы;
- 2) информационные системы;
- 3) информационные модели;
- 4) информационные технологии.

Правила техники безопасности и гигиены при работе на персональном компьютере

В наши дни персональный компьютер — неотъемлемая часть обеспечения работы, учебы и досуга. Проводя за ним долгие часы, мы не склонны задумываться, что перед нами не только бесценный помощник, но и потенциальный источник повышенной опасности. Этой опасности можно избежать, если соблюдать несложные правила.

Прежде всего не надо забывать, что компьютер — электроприбор, находящийся под напряжением, которое может быть смертельно опасным. Под опасным напряжением находятся и многие отдельные узлы системного блока, а также устройства, составляющие часть рабочего места человека, пользующегося персональным компьютером: монитор, принтер, сканер, звуковые колонки и т. д. В связи с этим недопустимо:

- вскрывать корпус компьютера, не отключенного от электрической сети;
- использовать некачественные или изношенные провода, электрические розетки, удлинители и иные вспомогательные устройства;
- располагать электрические провода таким образом, что на них можно наступить при ходьбе или случайно задеть телом при работе за компьютером.

Следует также знать, что на отдельных элементах компьютера (например, мониторах некоторых типов, источниках бесперебойного питания) высокое напряжение может сохраняться и некоторое время после отключения компьютера от сети.

Компьютер является и потенциальным источником пожарной опасности. Положенные на системный блок или источник питания листы бумаги, особенно если они закрывают вентиляционные отверстия, могут воспламениться (или привести к выходу

из строя компьютера за счет перегрева). В вентиляционных отверстиях узлов компьютера неизбежно скапливается пыль, и это тоже не способствует его качественной работе. Эту пыль надо хотя бы раз в год удалять с помощью пылесоса (особенно из системного блока, не вскрывая его).

Давайте не будем путать: из того, что вы умеете программировать, устанавливать и удалять программы, создавать презентации, общаться в Интернете и т. д., вовсе не следует, что вы являетесь квалифицированным специалистом по сборке-разборке компьютера. От правильной самооценки в этих вопросах зависит ваше здоровье (и даже жизнь).

Если правила техники безопасности при работе с компьютером, связанные с «электрической» составляющей, относительно просты и легковыполнимы, то правила личной гигиены в этой сфере (тоже не очень сложные) гораздо реже соблюдаются на практике, особенно когда мы не в школьном компьютерном классе, где за их соблюдением следит учитель, а дома. Вспомните, что элементарное требование «регулярно мойте руки мылом» предохраняет здоровье миллионов людей эффективнее самых совершенных лекарств, но многие люди болеют лишь потому, что его нарушили. При работе с компьютером имеет место аналогичная ситуация — люди, не соблюдая элементарные гигиенические нормы, теряют зрение, портят осанку, чрезмерно утомляются и даже доводят себя до психических расстройств. *Длительная работа за компьютером негативно сказывается на многих системах человеческого организма: эндокринной, иммунной и репродуктивной, на зрении и костно-мышечном аппарате.*

Обсудим некоторые проблемы в этой сфере детальнее.

Электромагнитные излучения, сопровождающие работу системного блока, монитора и других узлов компьютера, традиционно рассматриваются как первостепенные источники опасности для человека. Несмотря на то что современные персональные компьютеры в этом отношении гораздо менее опасны, чем их прародители, эта угроза сохраняется. Современные жидкокристаллические мониторы практически не дают опасного излучения в отличие от боковых и задних стенок системного блока и источников бесперебойного питания, которые поэтому не следует ставить вплотную к пользователю. В компьютерном классе, где стоит несколько компьютеров, системный блок соседнего компьютера не должен задней стенкой упираться в человека — на этот счет существуют строгие правила, как расставлять компьютеры.

Вредное влияние на зрение, оказываемое монитором, можно уменьшить как за счет высокого качества монитора, так и путем периодического выполнения несложных упражнений для глаз. Монитор должен удовлетворять международным стандартам безопасности, что фиксируется в сопроводительной документации. На него не должны падать блики от источников света; расстояние от экрана монитора до глаз пользователя должно составлять от 50 до 70 см. Не надо стремиться отодвинуть монитор как можно дальше, опасаясь вредных излучений, потому что для глаза важен также угол обзора наиболее характерных объектов. Оптимально размещение монитора на расстоянии примерно полутора размеров диагонали экрана. Важным параметром является частота кадров — она должна быть не менее 85 Гц. Периодически, почувствовав утомление зрения, необходимо сделать перерыв, отвести взгляд от экрана и сфокусировать его на удаленных предметах. Вредное влияние на зрение оказывает и недостаточная освещенность как помещения в целом, так и рабочего места в частности.

Вредное влияние на осанку, которое может привести к искривлению позвоночника и другим неприятностям, компенсируют правильным оснащением рабочего места — стола, стула, подставок для рук и ног. Рабочий стол и посадочное место должны иметь такую высоту, чтобы уровень глаз пользователя находился чуть выше центра монитора. Даже кратковременная работа с монитором, установленным слишком высоко, приводит к утомлению шейных отделов позвоночника. Клавиатура должна быть на такой высоте, чтобы пальцы рук располагались на ней свободно, без напряжения. При работе с мышью рука не должна находиться на весу. Локоть руки или хотя бы запястье должны иметь твердую опору.

И наконец, нельзя не сказать о **вредном влиянии на психику** некоторых пользователей персональных компьютеров. Число таких пользователей, к сожалению, возрастает. Компьютерная зависимость, проявляющаяся в том, что люди (прежде всего молодые) теряют интерес ко всему, кроме компьютерных игр и/или общения в Интернете, по классификации Всемирной организации здравоохранения отнесена к опасным психическим расстройствам. Эти люди постепенно теряют связь с действительностью и начинают жить в воображаемом мире. «Техника безопасности» здесь состоит в самоконтроле и контроле со стороны окружающих, помогающих вовремя оторваться от виртуального мира и вернуться в мир реальный.

Закончим этот обзор на оптимистичной ноте. Компьютеры и информационные технологии стали одним из величайших приобретений человечества за всё время развития науки и техники. Увы, практически любое техническое приобретение имеет и негативные стороны, но при соблюдении определенных правил можно минимизировать вредные последствия, и они несопоставимы с теми возможностями, которые компьютерная техника дала человечеству.

В работе с книгой вам помогут навигационные значки:

-  — важное утверждение или определение;
-  — вопросы и задания к параграфу;
-  — задания для подготовки к итоговой аттестации;
-  — практические работы на компьютере;
-  — межпредметные связи;
-  — групповая работа.
-  — К каждой главе учебника рекомендуется электронный образовательный ресурс (ЭОР) с сайта Федерального центра образовательных ресурсов (ФЦИОР): <http://fcior.edu.ru>.
-  — Проектное или исследовательское задание.
В ходе выполнения проекта (исследования) определите вид, в котором будут представлены его результаты. Можно выбрать один (или несколько) из следующих вариантов:
 - подготовить набор полезных ссылок с использованием веб-ресурсов;
 - подготовить небольшое выступление с использованием презентации (5–7 мин);
 - оформить доклад и поместить его на сайт школьной конференции;
 - подтвердить полученные результаты расчетами или графиками (диаграммами);
 - подготовить видеоролик;
 - разместить материалы проекта (исследования) в коллекции обучающих модулей по предмету на сайте школы.

Глава 1

ИНФОРМАЦИЯ

§ 1

Понятие информации

Наверное, самый сложный вопрос в информатике — это «Что такое информация?». На него нет однозначного ответа. Смысл этого понятия зависит от контекста (содержания разговора, текста), в котором оно употребляется.

В курсе информатики основной школы информация рассматривалась в разных контекстах. С позиции человека, информация — это содержание сообщений, это самые разнообразные сведения, которые человек получает из окружающего мира через свои органы чувств. Из совокупности получаемой человеком информации формируются его знания об окружающем мире и о себе самом.

Рассказывая о компьютере, мы говорили, что компьютер — это универсальный программно управляемый автомат для работы с информацией. В таком контексте не обсуждается смысл информации. Смысл — это значение, которое придает информации человек. Компьютер же работает с битами, с двоичными кодами. Вникать в их «смысл» компьютер не в состоянии. Поэтому правильнее называть информацию, циркулирующую в устройствах компьютера, **данными**. Тем не менее в разговорной речи, в литературе часто говорят о том, что компьютер хранит, обрабатывает, передает и принимает информацию. Ничего страшного в этом нет. Надо лишь понимать, что в «компьютерном контексте» понятие «информация» отождествляется с понятием «данные».

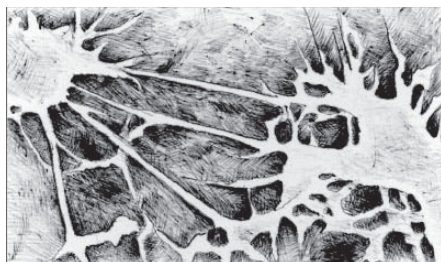
В толковом словаре В. И. Даля нет слова «информация». Термин «информация» начал широко употребляться с середины XX века. В наибольшей степени понятие информации обязано своим распространением двум научным направлениям: **теории связи** и **кибернетике**. Автор теории связи Клод Шеннон, анализируя технические системы связи: телеграф, телефон, радио, рассматривал их как *системы передачи информации*. В таких системах информация передается в виде после-



довательностей сигналов: электрических или электромагнитных. Развитие теории связи послужило созданию **теории информации**, решающей проблему измерения информации.

Основатель кибернетики Норберт Винер анализировал разнообразные процессы управления в живых организмах и в технических системах. Процессы управления рассматриваются в кибернетике как информационные процессы. *Информация в системах управления циркулирует в виде сигналов, передаваемых по информационным каналам.*

В XX веке понятие информации повсеместно проникает в науку. **Нейрофизиология** (раздел биологии) изучает механизмы нервной деятельности животного и человека. Эта наука строит модель информационных процессов, происходящих в организме. Поступающая извне информация превращается в сигналы электрохимической природы, которые от органов чувств передаются по нервным волокнам к нейронам (нервным клеткам) мозга. Мозг передает управляющую информацию в виде сигналов той же природы к мышечным тканям, управляя, таким образом, органами движения. Описанный механизм хорошо согласуется с кибернетической моделью Н. Винера.



В другой биологической науке — **генетике** используется понятие *наследственной информации*, заложенной в структуре молекул ДНК, присутствующих в ядрах клеток живых организмов (растений, животных). Генетика доказала, что эта структура является своеобразным кодом, определяющим функционирование всего организма: его рост, развитие, патологии и пр. Молекула ДНК передает наследственную информацию от поколения к поколению.

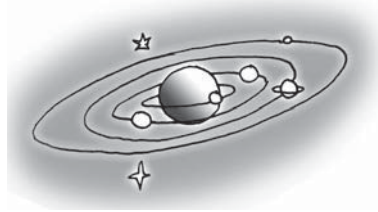
Понятие информации относится к числу фундаментальных, т. е. является основополагающим для науки и не объясняется через другие понятия. В этом смысле информация встает в один ряд с такими фундаментальными научными понятиями, как





вещество, энергия, пространство, время. Осмыслением информации как фундаментального понятия занимается наука философия.

Согласно одной из философских концепций, *информация является свойством всего сущего*, всех материальных объектов мира. Такая концепция информации называется **атрибутивной** (информация — атрибут всех материальных объектов). Информация в мире возникла вместе со Вселенной. С такой предельно широкой точки зрения, информация проявляется в воздействии одних объектов на другие, в изменениях, к которым такие воздействия приводят.



Другую философскую концепцию информации называют **функциональной**. Согласно функциональному подходу, *информация появилась лишь с возникновением жизни*, так как связана с функционированием сложных самоорганизующихся систем, к которым относятся живые организмы и человеческое общество. Можно еще сказать

так: информация — это атрибут, свойственный только живой природе. Это один из существенных признаков, отделяющих в природе живое от неживого.

Третья философская концепция информации — **антропоцентрическая**, согласно которой *информация существует лишь в человеческом сознании, в человеческом восприятии*. Информационная деятельность присуща только человеку, происходит в социальных системах. Создавая информационную технику, человек создает инструменты для своей информационной деятельности.



Делая выбор между различными точками зрения, надо помнить, что всякая научная теория — лишь модель бесконечно сложного мира, поэтому она не может отражать его точно и в полной мере.



Можно сказать, что употребление понятия «информация» в повседневной жизни происходит в антропоцентрическом контексте. Для любого из нас естественно воспринимать информацию как сообщения, которыми обмениваются люди. Например, средства массовой информации (СМИ) предназначены для распространения сообщений, новостей среди населения.



Система основных понятий

Понятие информации		
Философия	<i>Атрибутивная концепция:</i> информация — всеобщее свойство (атрибут) материи	
	<i>Функциональная концепция:</i> информация и информационные процессы присущи только живой природе, являются ее функцией	
	<i>Антропоцентрическая концепция:</i> информация и информационные процессы присущи только человеку	
Теория информации	Результат развития теории связи (К. Шеннон)	Информация — содержание, заложенное в знаковые (сигнальные) последовательности
Кибернетика	Исследует информационные процессы в системах управления (Н. Винер)	Информация — содержание сигналов, передаваемых по каналам связи в системах управления
Нейрофизиология	Изучает информационные процессы в механизмах нервной деятельности животного и человека	Информация — содержание сигналов электрохимической природы, передающихся по нервным волокнам организма
Генетика	Изучает механизмы наследственности, пользуется понятием «наследственная информация»	Информация — содержание генетического кода — структуры молекул ДНК, входящих в состав клетки живого организма

Вопросы и задания

1. Какие существуют основные философские концепции информации?
2. Какая, с вашей точки зрения, концепция является наиболее верной?
3. Благодаря развитию каких наук понятие информации стало широко употребляемым?
4. В каких биологических науках активно используется понятие информации?
5. Что такое наследственная информация?
6. К какой философской концепции, на ваш взгляд, ближе употребление понятия информации в генетике?
7. Если под информацией понимать только то, что распространяется через книги, рукописи, произведения искусства, средства массовой информации, то к какой философской концепции ее можно будет отнести?
8. Согласны ли вы, что понятие информации имеет контекстный смысл? Если да, то покажите это на примерах.

§ 2 Представление информации, языки, кодирование

Из курса основной школы вам известно:

- Историческое развитие человека, формирование человеческого общества связано с развитием речи, с появлением и распространением языков. Язык — это знаковая система для представления и передачи информации.
- Люди сохраняют свои знания в записях на различных носителях. Благодаря этому знания передаются не только в пространстве, но и во времени — от поколения к поколению.
- Языки бывают естественные, например русский, китайский, английский, и формальные, например математическая символика, нотная грамота, языки программирования.

Письменность и кодирование информации

Под словом «кодирование» понимают процесс представления информации в виде, удобном для ее хранения и/или передачи. Следо-

вательно, запись текста на естественном языке можно рассматривать как способ кодирования речи с помощью графических элементов (букв, иероглифов). Записанный текст является кодом, заключающим в себе содержание речи, т. е. информацию.

Процесс чтения текста — это обратный по отношению к письму процесс, при котором письменный текст преобразуется в устную речь. Чтение можно назвать **декодированием** письменного текста. Схематически эти два процесса изображены на рис. 1.1.

Схема на рис. 1.1 типична для всех процессов, связанных с передачей информации.

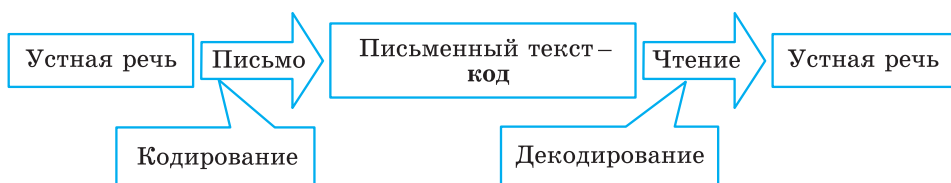


Рис. 1.1. Схема передачи информации с помощью письменности

Цели и способы кодирования

Теперь обратим внимание на то, что может существовать много способов кодирования одного и того же текста на одном и том же языке. Например, русский текст мы привыкли записывать с помощью русского алфавита. Но то же самое можно сделать, используя латинский алфавит. Иногда так приходится поступать, отправляя SMS по мобильному телефону, на котором нет русских букв, или электронное письмо на русском языке за границу, если у адресата нет русифицированного программного обеспечения. Например, фразу «Здравствуй, дорогой Саша!» приходится писать так: «Zdravstvui, dorogoi Sasha!».

Существует множество способов кодирования. Например, *стенография* — быстрый способ записи устной речи. Стенография появилась во времена, когда не существовало техники звукозаписи. Ею владели лишь немногие специально обученные люди — стенографисты. Они успевали записывать текст синхронно с речью выступающего человека. В стенограмме один значок обозначает целое слово или сочетание букв. Расшифровать (декодировать) стенограмму мог только сам стенографист.

Посмотрите на текст стенограммы на рис. 1.2. Там написано следующее: «Говорить умеют все люди на свете. Даже у самых примитивных племен есть речь. Язык — это нечто всеобщее и самое человеческое, что есть на свете».

Можно придумать и другие способы кодирования.

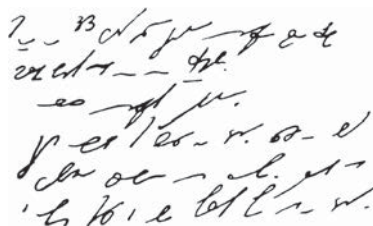


Рис. 1.2. Стенограмма

Приведенные примеры иллюстрируют следующее важное правило: *для кодирования одной и той же информации могут быть использованы разные способы; их выбор зависит от ряда обстоятельств: цели кодирования, условий, имеющихся средств.* Если надо записать текст в темпе речи, делаем это с помощью стенографии; если надо передать текст за границу, пользуемся латинским алфавитом; если надо представить текст в виде, понятном для грамотного русского человека, записываем его по правилам грамматики русского языка.

Еще одно важное обстоятельство: *выбор способа кодирования информации может быть связан с предполагаемым способом ее обработки.* Обсудим это на примере представления чисел — количественной информации. Используя русский алфавит, можно записать число «тридцать пять». Используя же алфавит арабской десятичной системы счисления, пишем: 35. Пусть вам надо произвести вычисления. Скажите, какая запись удобнее для выполнения расчетов: «тридцать пять умножить на сто двадцать семь» или « 35×127 »? Очевидно, что для перемножения многозначных чисел вы будете пользоваться второй записью.

Заметим, что эти две записи, эквивалентные по смыслу, используют разные языки: первая — естественный русский язык, вторая — формальный язык математики, не имеющий национальной принадлежности. Переход от представления на естественном языке к представлению на формальном языке можно также рассматривать как кодирование. Человеку удобно использовать для кодирования чисел десятичную систему счисления, а компьютеру — двоичную систему.

Широко используемыми в информатике формальными языками являются языки программирования.

В некоторых случаях возникает потребность засекречивания текста сообщения или документа, для того чтобы его не смогли



прочитать те, кому не положено. Это называется *защитой от несанкционированного доступа*. В таком случае секретный текст шифруется. В давние времена шифрование называлось тайнописью. **Шифрование** представляет собой процесс превращения открытого текста в зашифрованный, а **дешифрование** — процесс обратного преобразования, при котором восстанавливается исходный текст. Шифрование — это тоже кодирование, но с засекреченным методом, известным только источнику и адресату. Методами шифрования занимается наука **криптография**.

История технических способов кодирования информации

С появлением технических средств хранения и передачи информации возникли новые идеи и приемы кодирования. Первым техническим средством передачи информации на расстояние стал телеграф, изобретенный в 1837 году американцем Сэмюэлем Морзе. Телеграфное сообщение — это последовательность электрических сигналов, передаваемая от одного телеграфного аппарата по проводам к другому телеграфному аппарату. Эти технические обстоятельства привели Морзе к идее использования всего двух видов сигналов — короткого и длинного — для кодирования сообщения, передаваемого по линиям телеграфной связи.



**Сэмюэль Финли
Бриз Морзе**
(1791–1872), США

Такой способ кодирования получил название азбуки Морзе. В ней каждая буква алфавита кодируется последовательностью коротких сигналов (точек) и длинных сигналов (тире). Буквы отделяются друг от друга паузами — отсутствием сигналов.

В таблице на рис. 1.3 показана азбука Морзе применительно к русскому алфавиту. Специальных знаков препинания в ней нет. Их обычно записывают словами: «тчк» — точка, «зпт» — запятая и т. п.

Самым знаменитым телеграфным сообщением является сигнал бедствия «SOS» (Save Our Souls — спасите наши души). Вот как он выглядит в коде азбуки Морзе:

... — — — ...

Три точки обозначают букву S, три тире — букву O. Две паузы отделяют буквы друг от друга.

А . —	И ..	Р . — .	Ш — — — —
Б — ...	Й . — — —	С ...	Щ — — . —
В . — —	К — . —	Т —	Ъ . — — . — .
Г — — .	Л . — ..	У .. —	Ы — . — —
Д — ..	М — —	Ф .. — .	Ь — .. —
Е .	Н — .	Х	Э .. — ..
Ж ... —	О — — —	Ц — . — .	Ю .. — —
З — — ..	П . — — .	Ч — — — .	Я . — . —

Рис. 1.3. Кодовая таблица азбуки Морзе

Характерной особенностью азбуки Морзе является переменная длина кода разных букв, поэтому код Морзе называют *неравномерным кодом*. Буквы, которые встречаются в тексте чаще, имеют более короткий код, чем редкие буквы. Например, код буквы «Е» — одна точка, а код буквы «Ъ» состоит из шести знаков. Зачем так сделано? Чтобы сократить длину всего сообщения. Но из-за переменной длины кода букв возникает проблема отделения букв друг от друга в тексте. Поэтому приходится для разделения использовать паузу (пропуск). Следовательно, телеграфный алфавит Морзе является троичным, так как в нем используется три знака: точка, тире, пропуск.

Равномерный телеграфный код был изобретен французом Жаном Морисом Бодо в конце XIX века. В нем использовалось всего два вида сигналов. Неважно, как их назвать: точка и тире, плюс и минус, ноль и единица. Это два отличающихся друг от друга электрических сигнала.

В коде Бодо длина кодов всех символов алфавита одинакова и равна пяти. В таком случае не возникает проблемы отделения букв друг от друга: каждая пятерка сигналов — это знак текста.

Код Бодо — это первый в истории техники способ двоичного кодирования информации. Благодаря идее Бодо удалось



Жан Морис Эмиль Бодо (1845–1903),
Франция

автоматизировать процесс передачи и печати букв. Был создан клавишный телеграфный аппарат. Нажатие клавиши с определенной буквой вырабатывает соответствующий пятиимпульсный сигнал, который передается по линии связи. Принимающий аппарат под воздействием этого сигнала печатает ту же букву на бумажной ленте.

Из курса информатики основной школы вам известно, что в современных компьютерах для кодирования текстов также применяется равномерный двоичный код. Проблемы кодирования информации в компьютере и при передаче данных по сети мы рассмотрим несколько позже.

Система основных понятий

Представление информации					
Языки представления информации					
Естественные: русский, китайский, английский и др.			Формальные: язык математики, нотная грамота, языки программирования и др.		
Кодирование					
Цели кодирования					
Засекречивание информации	Быстрый способ записи	Передача по техническим каналам связи		Выполнение математических вычислений	
Шифрование	Стенография	Телеграфный код		Системы счисления	
Алгоритмы криптографии	Один знак — слово или сочетание букв	Код Морзе: неравномерный, троичный код	Код Бодо: равномерный, двоичный код	Для человека: десятичная с. с.	Для компьютера: двоичная с. с.



Вопросы и задания

1. Чем отличаются естественные языки от формальных?
2. Как вы думаете, латынь — это естественный или формальный язык?
3. С каким формальным языком программирования вы знакомы? Для чего он предназначен?
4. Что такое кодирование и декодирование?
5. От чего может зависеть способ кодирования?
6. В чем преимущество кода Бодо по сравнению с кодом Морзе?
7. В чем преимущество кода Морзе по сравнению с кодом Бодо?

§ 3

Измерение информации. Алфавитный подход

Вопрос об измерении количества информации является очень важным как для науки, так и для практики. В самом деле, если информация является предметом нашей деятельности, мы ее храним, передаем, принимаем, обрабатываем. Поэтому важно договориться о способе ее измерения, позволяющем, например, ответить на вопросы: достаточно ли места на носителе, чтобы разместить нужную нам информацию, или сколько времени потребуется, чтобы передать ее по имеющемуся каналу связи? Величина, которая нас в этих ситуациях интересует, называется **объемом информации**. В таком случае говорят об **алфавитном**, или **объемном**, подходе к измерению информации.

Алфавитный подход к измерению информации применяется в цифровых (компьютерных) системах хранения и передачи информации. В этих системах используется двоичный способ кодирования информации. При алфавитном подходе для определения количества информации имеет значение лишь размер (объем) хранимого и передаваемого кода. Алфавитный подход еще называют **объемным подходом**. Из курса информатики 7–9 классов вы знаете, что если с помощью i -разрядного двоичного кода можно закодировать алфавит, состоящий из N символов (где N — целая степень двойки), то эти величины связаны между собой по формуле:

$$2^i = N.$$

Число N называется **мощностью алфавита**

Если, например, $i = 2$, то можно построить 4 двухразрядные комбинации из нулей и единиц, т. е. закодировать 4 символа. При $i = 3$ существует 8 трехразрядных комбинаций нулей и единиц (кодируется 8 символов):

$i = 2$:	00	01	10	11					
$i = 3$:	000	001	010	011	100	101	110	111	

Английский алфавит содержит 26 букв. Для записи текста нужны еще как минимум шесть символов: пробел, точка, запятая, вопросительный знак, восклицательный знак, тире. В сумме получается расширенный алфавит мощностью в 32 символа.

Поскольку $32 = 2^5$, все символы можно закодировать всевозможными пятиразрядными двоичными кодами от 00000 до 11111. Именно пятиразрядный код использовался в телеграфных аппаратах, появившихся еще в XIX веке. Телеграфный аппарат при вводе переводил английский текст в двоичный код, длина которого в 5 раз больше, чем длина исходного текста.



В двоичном коде каждая двоичная цифра несет одну единицу информации, которая называется 1 бит.



Бит является основной единицей измерения информации.

Длина двоичного кода, с помощью которого кодируется символ алфавита, называется **информационным весом символа**. В рассмотренном выше примере информационный вес символа расширенного английского алфавита оказался равным 5 битам.

Информационный объем текста складывается из информационных весов всех составляющих текст символов. Например, английский текст из 1000 символов в телеграфном сообщении будет иметь информационный объем 5000 битов.

Алфавит русского языка включает 33 буквы. Если к нему добавить еще пробел и пять знаков препинания, то получится набор из 39 символов. Для двоичного кодирования символов такого алфавита пятиразрядного кода уже недостаточно. Нужен как минимум 6-разрядный код. Поскольку $2^6 = 64$, остается еще

резерв для 25 символов ($64 - 39 = 25$). Его можно использовать для кодирования цифр, всевозможных скобок, знаков математических операций и других символов, встречающихся в русском тексте. Следовательно, информационный вес символа в расширенном русском алфавите будет равен 6 битам. А текст из 1000 символов будет иметь объем 6000 битов.

Итак, если i — информационный вес символа алфавита, а K — количество символов в тексте, записанном с помощью этого алфавита, то **информационный объем I текста** выражается формулой:

$$I = K \times i \text{ (битов).}$$

Идея измерения количества информации в сообщении через длину двоичного кода этого сообщения принадлежит выдающемуся российскому математику Андрею Николаевичу Колмогорову. Согласно Колмогорову, количество информации, содержащееся в тексте, определяется минимально возможной длиной двоичного кода, необходимого для представления этого текста.

Для определения информационного веса символа полезно знать ряд целых степеней двойки. Вот как он выглядит в диапазоне от 2^1 до 2^{10} :



А. Н. Колмогоров
(1903–1987),
Россия

i	1	2	3	4	5	6	7	8	9	10
2^i	2	4	8	16	32	64	128	256	512	1024

Поскольку мощность N алфавита может не являться целой степенью двойки, информационный вес символа алфавита мощности N определяется следующим образом. Находится ближайшее к N значение во второй строке таблицы, не меньшее чем N . Соответствующее значение i в первой строке будет равно информационному весу символа.

Пример. Определим информационный вес символа алфавита, включающего в себя все строчные и прописные русские буквы (66); цифры (10); знаки препинания, скобки, кавычки (10). Всего получается 86 символов.

Поскольку $2^6 < 86 < 2^7$, информационный вес символов данного алфавита равен 7 битам. Это означает, что все 86 символов можно закодировать семиразрядными двоичными кодами.

Для двоичного представления текстов в компьютере чаще всего применяется восьмиразрядный код. С помощью восьмиразрядного кода можно закодировать алфавит из 256 символов, поскольку $256 = 2^8$. В стандартную кодовую таблицу (например, используемую в ОС Windows таблицу ANSI) помещаются все необходимые символы: английские и русские буквы — прописные и строчные, цифры, знаки препинания, знаки арифметических операций, всевозможные скобки и пр.

Более крупной, чем бит, единицей измерения информации является **байт**: *1 байт = 8 битов*.



Информационный объем текста в памяти компьютера измеряется в байтах. Он равен количеству символов в записи текста.

Одна страница текста на листе формата А4 кегля 12 с одинарным интервалом между строками в компьютерном представлении будет иметь объем 4000 байтов, так как на ней помещается примерно 4000 знаков.

Помимо бита и байта, для измерения информации используются и более крупные единицы:

1 Кб (килобайт)	= 2^{10} байтов	= 1024 байта;
1 Мб (мегабайт)	= 2^{10} Кб	= 1024 Кб;
1 Гб (гигабайт)	= 2^{10} Мб	= 1024 Мб;
1 Тб (терабайт)	= 2^{10} Гб	= 1024 Гб.

Объем той же страницы текста будет равен приблизительно 3,9 Кб. А книга из 500 таких страниц займет в памяти компьютера примерно 1,9 Мб.

В компьютере любые виды информации: тексты, числа, изображения, звуки — представляются в форме двоичного кода.



Объем информации любого вида, выраженный в битах, равен длине двоичного кода, в котором эта информация представлена.

Система основных понятий



Измерение информации. Алфавитный (объемный) подход				
Применяется в цифровых системах хранения и передачи информации				
<i>Объем информации равен длине двоичного кода</i> Основная единица: 1 бит — один разряд двоичного кода				
Информационный вес символа (i битов) алфавита мощностью N определяется из уравнения: $2^i = M$, где M — ближайшая к N сверху целая степень двойки		Информационный объем I текста, содержащего K символов: $I = K \cdot i$ битов, где i — информационный вес одного символа		
Производные единицы				
Байт 1 байт = 8 битов	Килобайт (Кб) 1 Кб = 1024 байта	Мегабайт (Мб) 1 Мб = 1024 Кб	Гигабайт (Гб) 1 Гб = 1024 Мб	Терабайт (Тб) 1 Тб = 1024 Мб

Вопросы и задания



1. Есть ли связь между алфавитным подходом к измерению информации и содержанием информации?
2. В чем можно измерить объем письменного или печатного текста?
3. Оцените объем одной страницы данного учебника в байтах.
4. Что такое бит с позиции алфавитного подхода к измерению информации?
5. Как определяется информационный объем текста по А. Н. Колмогорову?
6. Какой информационный вес имеет каждая буква русского алфавита?
7. Какие единицы используются для измерения объема информации на компьютерных носителях?
8. Сообщение, записанное буквами из 64-символьного алфавита, содержит 100 символов. Какой объем информации оно несет?
9. Сколько символов содержит сообщение, записанное с помощью 16-символьного алфавита, если его объем составляет $1/16$ Мб?
10. Сообщение занимает 2 страницы и содержит $1/16$ Кб информации. На каждой странице 256 символов. Какова мощность используемого алфавита?





11. Возьмите страницу текста из данного учебника и подсчитайте информационные объемы текста, получаемые при кодировании его семиразрядным и восьмиразрядным кодами. Результаты выразите в килобайтах и мегабайтах.

§ 4

Измерение информации. Содержательный подход

Неопределенность знания и количество информации

Содержательный подход к измерению информации отталкивается от определения информации как содержания сообщения, получаемого человеком. Сущность содержательного подхода заключается в следующем: сообщение, информирующее об исходе какого-то события, снимает неопределенность знания человека об этом событии.

Чем больше первоначальная неопределенность знания, тем больше информации несет сообщение, снимающее эту неопределенность.

Приведем примеры, иллюстрирующие данное утверждение.

Ситуация 1. В ваш класс назначен новый учитель информатики; на вопрос «Это мужчина или женщина?» вам ответили: «Мужчина».

Ситуация 2. На чемпионате страны по футболу играли команды «Динамо» и «Зенит». Из спортивных новостей по радио вы узнаете, что игра закончилась победой «Зенита».

Ситуация 3. На выборах мэра города было представлено четыре кандидата. После подведения итогов голосования вы узнали, что избран Н. Н. Никитин.

Вопрос: в какой из трех ситуаций полученное сообщение несет больше информации?

Неопределенность знания — это количество возможных вариантов ответа на интересовавший вас вопрос. Еще можно сказать: возможных исходов события. Здесь событие — например, выборы мэра; исход — выбор, например, Н. Н. Никитина.

В первой ситуации 2 варианта ответа: мужчина, женщина; во второй ситуации 3 варианта: выиграл «Зенит», ничья, выиграло

«Динамо»; в третьей ситуации — 4 варианта: 4 кандидата на пост мэра.

Согласно данному выше определению, наибольшее количество информации несет сообщение в третьей ситуации, поскольку неопределенность знания об исходе события в этом случае была наибольшей.

В 40-х годах XX века проблема измерения информации была решена американским ученым Клодом Шенноном — основателем *теории информации*. Согласно Шеннону, **информация — это снятая неопределенность** знания человека об исходе какого-то события.



Клод Элвуд
Шеннон
(1916–2001)

В теории информации единица измерения информации определяется следующим образом.

Сообщение, уменьшающее неопределенность знания об исходе некоторого события в два раза, несет 1 бит информации.



Согласно этому определению, сообщение в первой из описанных ситуаций несет 1 бит информации, поскольку из двух возможных вариантов ответа был выбран один.

Следовательно, количество информации, полученное во второй и в третьей ситуациях, больше, чем один бит. Но как измерить это количество?

Рассмотрим еще один пример.

Ученик написал контрольную по информатике и спрашивает учителя о полученной оценке. Оценка может оказаться любой: от 2 до 5. На что учитель отвечает: «Угадай оценку за два вопроса, ответом на которые может быть только "да" или "нет"». Подумав, ученик задал первый вопрос: «Оценка выше тройки?». «Да», — ответил учитель. Второй вопрос: «Это пятерка?». «Нет», — ответил учитель. Ученик понял, что он получил четверку. Какая бы ни была оценка, таким способом она будет угадана!

Первоначально неопределенность знания (количество возможных оценок) была равна четырем. С ответом на каждый вопрос неопределенность знания уменьшалась в 2 раза и, следовательно, согласно данному выше определению, передавался 1 бит информации.

Первоначальные варианты:

2	3	4	5
---	---	---	---

Варианты, оставшиеся после 1-го вопроса:
(1 бит)

		4	5
--	--	---	---

Вариант, оставшийся после 2-го вопроса:
(+ 1 бит)

		4	
--	--	---	--

Узнав оценку (одну из четырех возможных), ученик получил 2 бита информации.

Рассмотрим еще один частный пример, а затем выведем общее правило.

Вы едете на электропоезде, в котором 8 вагонов, а на вокзале вас встречает товарищ. Товарищ позвонил вам по мобильному телефону и спросил, в каком вагоне вы едете. Вы предлагаете угадать номер вагона, задав наименьшее количество вопросов, ответами на которые могут быть только слова «да» или «нет».

Немного подумав, товарищ стал спрашивать:

- Номер вагона больше четырех?
- Да.
- Номер вагона больше шести?
- Нет.
- Это шестой вагон?
- Нет.
- Ну теперь все ясно! Ты едешь в пятом вагоне!

Схематически поиск номера вагона выглядит так:

Первоначальные варианты:

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

После 1-го вопроса (1 бит):

				5	6	7	8
--	--	--	--	---	---	---	---

После 2-го вопроса (+ 1 бит):

				5	6		
--	--	--	--	---	---	--	--

После 3-го вопроса (+ 1 бит):

				5			
--	--	--	--	---	--	--	--

Каждый ответ уменьшал неопределенность знания в два раза. Всего было задано три вопроса. Значит, в сумме набрано 3 бита информации. То есть сообщение о том, что вы едете в пятом вагоне, несет 3 бита информации.

Способ решения проблемы, примененный в примерах с оценками и вагонами, называется *методом половинного деления*: ответ на каждый вопрос уменьшает неопределенность знания, имеющуюся перед ответом на этот вопрос, наполовину. Каждый такой ответ несет 1 бит информации.

Заметим, что решение подобных проблем методом половинного деления наиболее рационально. Таким способом всегда можно угадать, например, любой из восьми вариантов за 3 вопроса. Если бы поиск производился последовательным перебором: «Ты едешь в первом вагоне?» «Нет», «Во втором вагоне?» «Нет» и т. д., то про пятый вагон вы смогли бы узнать после пяти вопросов, а про восьмой — после восьми.

«Главная формула» информатики

Сформулируем одно очень важное условие, относящееся к рассмотренным примерам. Во всех ситуациях предполагается, что *все возможные исходы события равновероятны*. Равновероятно, что учитель может быть мужчиной или женщиной; равновероятен любой исход футбольного матча, равновероятен выбор одного из четырех кандидатов в мэры города. То же относится и к примерам с оценками и вагонами.

Тогда полученные нами результаты описываются следующими формулировками:

- сообщение об одном из двух равновероятных исходов некоторого события несет 1 бит информации;
- сообщение об одном из четырех равновероятных исходов некоторого события несет 2 бита информации;
- сообщение об одном из восьми равновероятных исходов некоторого события несет 3 бита информации.

Обозначим буквой N количество возможных исходов события, или, как мы это еще называли, — неопределенность знания. Буквой i будем обозначать количество информации в сообщении об одном из N результатов.

В примере с учителем: $N = 2$, $i = 1$ бит;

в примере с оценками: $N = 4$, $i = 2$ бита;

в примере с вагонами: $N = 8$, $i = 3$ бита.

Нетрудно заметить, что связь между этими величинами выражается следующей формулой:

$$2^i = N.$$

Действительно: $2^1 = 2$; $2^2 = 4$; $2^3 = 8$.

С полученной формулой вы уже знакомы из курса информатики для 7 класса и еще не однажды с ней встретитесь. Значение этой формулы столь велико, что мы называли ее **главной формулой информатики**. Если величина N известна, а i неизвестно, то данная формула становится уравнением для определения i . В математике такое уравнение называется *показательным уравнением*.

Пример. Вернемся к рассмотренному выше примеру с вагонами. Пусть в поезде не 8, а 16 вагонов. Чтобы ответить на вопрос, какое количество информации содержится в сообщении о номере искомого вагона, нужно решить уравнение:

$$2^i = 16.$$

Поскольку $16 = 2^4$, $i = 4$ бита.

Количество информации i , содержащееся в сообщении об одном из N равновероятных исходов некоторого события, определяется из решения показательного уравнения:

$$2^i = N.$$

Пример. В кинозале 16 рядов, в каждом ряду 32 места. Какое количество информации несет сообщение о том, что вам купили билет на 12-й ряд, 10-е место?

Решение задачи: в кинозале всего $16 \cdot 32 = 512$ мест. Сообщение о купленном билете однозначно определяет выбор одного из этих мест. Из уравнения $2^i = 512 = 2^9$ получаем: $i = 9$ битов.

Но эту же задачу можно решать иначе. Сообщение о номере ряда несет 4 бита информации, так как $2^4 = 16$. Сообщение о номере места несет 5 битов информации, так как $2^5 = 32$. В целом сообщение про ряд и место несет: $4 + 5 = 9$ битов информации.

Данный пример иллюстрирует выполнение *закона аддитивности количества информации* (правило сложения): *количество информации в сообщении одновременно о нескольких результатах независимых друг от друга событий равно сумме количеств информации о каждом событии отдельно.*

Сделаем одно важное замечание. С формулой $2^i = N$ мы уже встречались, обсуждая алфавитный подход к измерению информации (см. § 3). В этом случае N рассматривалось как мощность алфавита, а i — как информационный вес каждого символа алфавита. Если допустить, что все символы алфавита появляются в тексте с одинаковой частотой, т. е. равновероятно, то информационный вес символа i тождественен количеству информации в сообщении о появлении любого символа в тексте. При этом N — неопределенность знания о том, какой именно символ алфавита должен стоять в данной позиции текста. Данный факт демонстрирует связь между алфавитным и содержательным подходами к измерению информации.

Формула Хартли

Если значение N равно целой степени двойки (4, 8, 16, 32, 64 и т. д.), то показательное уравнение легко решить в уме, поскольку i будет целым числом. А чему равно количество информации в сообщении о результате матча «Динамо»–«Зенит»? В этой ситуации $N = 3$. Можно догадаться, что решение уравнения

$$2^i = 3$$

будет дробным числом, лежащим между 1 и 2, поскольку $2^1 = 2 < 3$, а $2^2 = 4 > 3$. А как точнее узнать это число?

В математике существует функция, с помощью которой решается показательное уравнение. Эта функция называется *логарифмом*, и решение нашего уравнения записывается следующим образом:

$$i = \log_2 N.$$

Читается это так: «логарифм от N по основанию 2». Смысл очень простой: логарифм по основанию 2 от N — это степень, в которую нужно возвести 2, чтобы получить N . Например, вычисление уже известных вам значений можно представить так:

$$\log_2 2 = 1, \quad \log_2 4 = 2, \quad \log_2 8 = 3.$$

Значения логарифмов находятся с помощью специальных логарифмических таблиц. Также можно использовать инженерный калькулятор или табличный процессор. Определим количество информации, полученной из сообщения об одном исходе события из трех равновероятных, с помощью электронной таблицы. На рисунке 1.4 представлены два режима электронной таблицы: режим отображения формул и режим отображения значений.



	A	B
1	N	i (битов)
2	3	=LOG(A2;2)

	A	B
1	N	i (битов)
2	3	1,584962501

Рис. 1.4. Определение количества информации в электронных таблицах с помощью функции логарифма



Ральф Хартли
(1888–1970)

В табличном процессоре Microsoft Excel функция логарифма имеет следующий вид: LOG(аргумент; основание). Аргумент — значение N находится в ячейке A2, а основание логарифма равно 2. В результате получаем с точностью до девяти знаков после запятой:

$$i = \log_2 3 = 1,584962501 \text{ (бита).}$$

Формула для измерения количества информации: $i = \log_2 N$ была предложена американским ученым Ральфом Хартли — одним из основоположников теории информации.

Формула Хартли:

$$i = \log_2 N.$$

Здесь i — количество информации, содержащееся в сообщении об одном из N равновероятных исходов события.

Данный пример показал, что количество информации, определяемое с использованием содержательного подхода, может быть дробной величиной, в то время как информационный объем, вычисляемый путем применения алфавитного подхода, может иметь только целочисленное значение.

Система основных понятий



Измерение информации. Содержательный подход	
Измеряется количество информации в сообщении об исходе некоторого события	
Равновероятные исходы: никакой результат не имеет преимущества перед другими	
Неопределенность знания — количество возможных исходов события (вариантов сообщения) — N	Количество информации в сообщении об одном исходе события — i битов
$2^i = N$	
Частный случай: два равновероятных результата события	
$N = 2$	$i = 1$ бит
1 бит — количество информации в сообщении об одном из двух равновероятных результатов некоторого события	
Формула Хартли: $i = \log_2 N$	

Вопросы и задания



1. Что такое неопределенность знания об исходе некоторого события?
2. Как определяется единица измерения количества информации в рамках содержательного подхода?
3. Придумайте несколько ситуаций, при которых сообщение несет 1 бит информации.
4. В каких случаях и по какой формуле можно вычислить количество информации, содержащейся в сообщении, используя содержательный подход?
5. Сколько битов информации несет сообщение о том, что из колоды в 32 карты достали «даму пик»?
6. При угадывании методом половинного деления целого числа из диапазона от 1 до N был получен 1 байт информации. Чему равно N ?
7. Проводятся две лотереи: «4 из 32» и «5 из 64». Сообщение о результатах какой из лотерей несет больше информации?





8. Используя формулу Хартли и электронные таблицы, определите количество информации в сообщениях о равновероятных событиях:
- на шестигранном игральном кубике выпала цифра 3;
 - в следующем году ремонт в школе начнется в феврале;
 - я приобрел абонемент в бассейн на среду;
 - из 30 учеников класса дежурить в школьной столовой назначили Дениса Скворцова.
9. Используя закон аддитивности количества информации, решите задачу о билете в кинотеатр со следующим дополнительным условием: в кинотеатре 4 зала. В билете указан номер зала, номер ряда и номер места. Какое количество информации заключено в билете?



§ 5

Представление чисел в компьютере

Главные правила представления данных в компьютере

Если бы мы могли заглянуть в содержание компьютерной памяти, то увидели бы там примерно то, что условно изображено на рис. 1.5.

1	1	0	0	1	1	1	1	0	0	0	1	0	1	0	0
0	1	0	0	1	1	1	0	1	0	0	0	1	1	1	1
1	1	0	1	1	0	1	1	0	1	1	0	0	1	1	1
1	1	0	0	1	0	1	0	0	0	1	1	0	1	1	1
1	1	0	0	0	1	1	0	1	0	0	0	1	0	1	0
1	1	1	0	0	1	0	1	0	1	1	1	0	0	0	1
1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	0
1	1	0	1	0	1	0	1	1	0	0	1	1	1	0	0
0	1	0	1	0	0	1	1	1	1	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1	1	0	1	0	1	0
1	0	0	0	1	1	0	1	0	1	0	1	0	1	0	0
1	1	0	0	1	1	1	0	0	1	1	0	1	0	1	1
0	0	1	1	1	0	0	1	1	0	1	0	1	0	1	0
0	1	1	1	1	0	1	0	1	0	1	0	0	1	1	1
0	0	1	1	1	0	1	0	1	0	1	0	1	1	1	0
1	1	1	1	1	0	1	0	1	1	1	0	0	1	1	1

Рис. 1.5. Образ компьютерной памяти

Рисунок 1.5 отражает известное вам еще из курса информатики основной школы правило представления данных, которое назовем **правилом № 1: данные (и программы) в памяти компьютера хранятся в двоичном виде**, т. е. в виде цепочек единиц и нулей.

Современный компьютер может хранить и обрабатывать данные, представляющие информацию четырех видов: числовую, текстовую, графическую, звуковую. Двоичный код, изображенный на рис. 1.5, может относиться к любому виду данных.

Правило № 2: *представление данных в компьютере дискретно.*

Правило № 3: *множество представимых в памяти компьютера величин ограничено и конечно.*

Представление чисел

Сначала поясним на образном примере, что такое дискретность.

Дискретное множество состоит из отделенных друг от друга элементов. Например, песок дискретен, поскольку он состоит из отдельных песчинок. А вода или масло непрерывны (в рамках наших ощущений, поскольку отдельные молекулы мы всё равно ощутить не можем). Этот пример нужен нам только для аналогии. Здесь мы не станем углубляться в изучение материального мира, а вернемся к предмету изучения информатики — информации.

Самым традиционным видом данных, с которым работают компьютеры, являются числа. ЭВМ первого поколения умели решать только математические задачи. Люди начали работать с числами еще с первобытных времен. Первоначально человек оперировал лишь целыми положительными (натуральными) числами: 1, 2, 3, 4, Очевидно, что *натуральный ряд — это дискретное множество чисел.*

В математике ряд *натуральных чисел бесконечен и не ограничен*. С появлением в математике понятия отрицательного числа (Р. Декарт, XVII век в Европе; в Индии значительно раньше) оказалось, что множество целых чисел не ограничено как «справа», так и «слева». Покажем это на числовой оси (рис. 1.6), символы ∞ обозначают бесконечность.

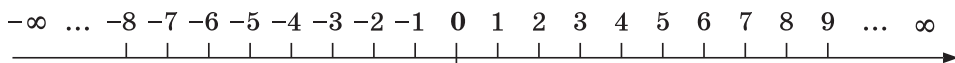


Рис. 1.6. Математическое множество целых чисел на числовой оси

Из сказанного следует вывод: *множество целых чисел в математике дискретно и не ограничено*. Отметим еще один факт: разность соседних чисел натурального ряда (данного и предыдущего) всегда равна единице. Эту величину назовем *шагом числовой последовательности*.

Любое вычислительное устройство (компьютер, калькулятор) может работать только с *ограниченным* множеством целых чисел. Возьмите в руки калькулятор, на индикаторном табло которого помещается 10 знаков. Самое большое положительное число, которое на него поместится:

	9	9	9	9	9	9	9	9	9
--	---	---	---	---	---	---	---	---	---

Самое большое по абсолютной величине (модулю) отрицательное число:

–	9	9	9	9	9	9	9	9	9
---	---	---	---	---	---	---	---	---	---

Аналогично дело обстоит и в компьютере.

Целые числа в компьютере

Правило № 4: *в памяти компьютера числа хранятся в двоичной системе счисления¹⁾*. С двоичной системой счисления вы знакомы из курса информатики 7–9 классов. Например, если под целое число выделяется ячейка памяти размером в 16 битов, то самое большое целое положительное число будет таким:

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

В десятичной системе счисления оно равно:

$$2^{15} - 1 = 32\,767.$$

¹⁾ Конечно, и «внутри калькулятора» числа представляются в двоичном виде. Однако мы в это вдаваться не будем, рассмотрев лишь внешнее представление. Пример с калькулятором нам нужен был только для иллюстрации проблемы ограниченности.

Здесь первый бит играет роль знака числа. Ноль — признак положительного числа. Самое большое по модулю отрицательное число равно $-32\,768$. Напомним (это было в курсе информатики основной школы), как получить его внутреннее представление:

- 1) перевести число $32\,768$ в двоичную систему счисления; это легко, поскольку $32\,768 = 2^{15}$:

1000000000000000;

- 2) инвертировать этот двоичный код, т. е. заменить нули на единицы, а единицы — на нули:

0111111111111111;

- 3) прибавить единицу к этому двоичному числу (складывать надо по правилам двоичной арифметики), в результате получим:

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Единица в первом бите обозначает знак «минус». Не нужно думать, что полученный код — это «минус ноль». Этот код представляет число $-32\,768$. Таковы правила машинного представления целых чисел. Данное представление называется **дополнительным кодом**.

Если под целое число в памяти компьютера отводится N битов, то диапазон значений целых чисел:

$$[-2^{N-1}, 2^{N-1} - 1],$$

т. е. ограниченность целого числа в компьютере возникает из-за ограничений на размер ячейки памяти. Отсюда же следует и конечность множества целых чисел.

Мы рассмотрели **формат представления целых чисел со знаком**, т. е. положительных и отрицательных. Бывает, что нужно работать только с положительными целыми числами. В таком случае используется **формат представления целых чисел без знака**. В этом формате самое маленькое число — ноль (все биты — нули), а самое большое число для 16-разрядной ячейки:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

В десятичной системе это $2^{16} - 1 = 65\,535$, примерно в два раза больше по модулю, чем в представлении со знаком.

Из всего сказанного делаем вывод: *целые числа в памяти компьютера — это дискретное, ограниченное и конечное множество.*

Границы множества целых чисел зависят от размера выделяемой ячейки памяти под целое число, а также от формата: со знаком или без знака. Шаг в компьютерном представлении последовательности целых чисел, как и в математическом, остается равным единице.

Рисунок 1.7 отражает то обстоятельство, что при переходе от математического представления множества целых чисел к представлению, используемому в информатике (компьютере), происходит переход к ограниченности и конечности.

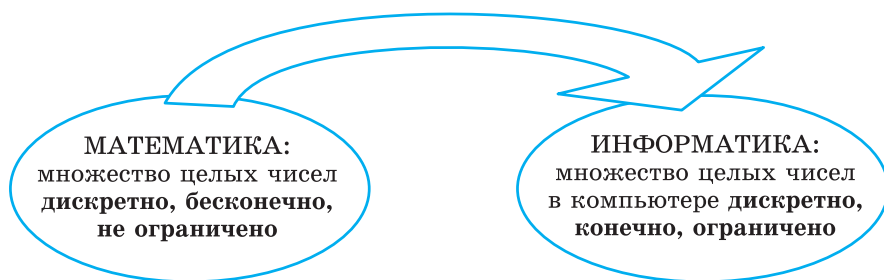


Рис. 1.7. Представление о множестве целых чисел в математике и в информатике

Вещественные числа в компьютере

Понятие вещественного (действительного) числа в математику ввел Исаак Ньютон в XVIII веке. В математике множество вещественных чисел непрерывно, бесконечно и не ограничено. Оно включает в себя множество целых чисел и еще бесконечное множество нецелых чисел. Между двумя любыми точками на числовой оси лежит бесконечное множество вещественных чисел, что и означает непрерывность множества.

Как мы говорили выше, числа в компьютере (в том числе и вещественные) представлены в двоичной системе счисления. Покажем, что множество вещественных чисел в компьютере

дискретно, ограничено и конечно. Нетрудно догадаться, что это, так же как и в случае целых чисел, вытекает из ограничения размера ячейки памяти.

Снова для примера возьмем калькулятор с десятиразрядным индикаторным табло. Экспериментально докажем дискретность представления вещественных чисел. Выполним на калькуляторе деление 1 на 3. Из математики вам известно, что $1/3$ — это рациональная дробь, представление которой в виде десятичной дроби содержит бесконечное количество цифр: $0,3333333333\dots$ (3 в периоде). На табло калькулятора вы увидите:

	0.	3	3	3	3	3	3	3	3
--	----	---	---	---	---	---	---	---	---

Первый разряд зарезервирован под знак числа. После запятой сохраняется 8 цифр, а остальные не вмещаются в *разрядную сетку* (так это обычно называют). Значит, это не точное значение, равное $1/3$, а его «урезанное» значение.

Следующее по величине число, которое помещается в разрядную сетку:

	0.	3	3	3	3	3	3	3	4
--	----	---	---	---	---	---	---	---	---

Оно больше предыдущего на $0,00000001$. Это шаг числовой последовательности. Следовательно, два рассмотренных числа разделены между собой конечным отрезком. Очевидно, что предыдущее число такое:

	0.	3	3	3	3	3	3	3	2
--	----	---	---	---	---	---	---	---	---

Оно тоже отделено от своего «соседа справа» по числовой оси шагом $0,00000001$. Отсюда делаем вывод: *множество вещественных чисел, представимых в калькуляторе, дискретно, поскольку числа отделены друг от друга конечными отрезками.*

А теперь выясним вот что: будет ли шаг в последовательности вещественных чисел на калькуляторе постоянной величиной (как у целых чисел)?

Вычислим выражение $100000/3$. Получим:

	3	3	3	3	3.	3	3	3	3
--	---	---	---	---	----	---	---	---	---

Это число в 100 000 раз больше предыдущего и, очевидно, тоже приближенное. Легко понять, что следующее вещественное число, которое можно получить на табло калькулятора, будет больше данного на 0,0001. Шаг стал гораздо больше.

Отсюда приходим к выводу: *множество вещественных чисел, представимых в калькуляторе, дискретно с переменной величиной шага между соседними числами.*

Если отметить на числовой оси точные значения вещественных чисел, которые представимы в калькуляторе, то эти точки будут расположены вдоль оси неравномерно. Ближе к нулю — чаще, дальше от нуля — реже (рис. 1.8).

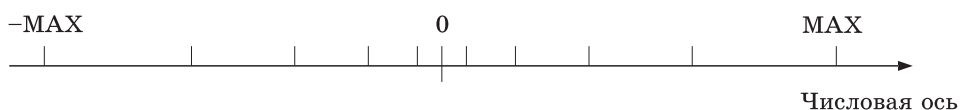


Рис. 1.8. Условное изображение взаимного расположения множества вещественных чисел, представимых в компьютере

Все выводы, которые мы делаем на примере калькулятора, полностью переносятся на компьютер с переходом к двоичной системе счисления и с учетом размера ячейки компьютера, отводимой под вещественные числа. Неравномерное расположение вещественных чисел, представимых в компьютере, также имеет место.

Ответим на вопрос: ограничено ли множество вещественных чисел в памяти компьютера? Если продолжать эксперименты с калькулятором, то ответ на этот вопрос будет таким: да, множество вещественных чисел в калькуляторе ограничено. Причиной тому служит все та же ограниченность разрядной сетки. Отсюда же следует и конечность множества.

Самое большое число у разных калькуляторов может оказаться разным. У самого простого это будет то же число, что мы получали раньше: 99999999. Если прибавить к нему единицу, то калькулятор выдаст сообщение об ошибке. А на другом, более «умном» и дорогом, калькуляторе прибавление единицы приведет к такому результату:

					1	e	+	0	9
--	--	--	--	--	---	---	---	---	---

Данную запись на табло надо понимать так: $1 \cdot 10^9$.

Такой формат записи числа называется **форматом с плавающей запятой**, в отличие от всех предыдущих примеров, где рассматривалось представление чисел в **формате с фиксированной запятой**.

Число, стоящее перед буквой «е», называется **мантиссой**, а стоящее после — **порядком**. «Умный» калькулятор перешел к представлению чисел в формате с плавающей запятой после того, как под формат с фиксированной запятой не стало хватать места на табло.

В компьютере то же самое: числа могут представляться как в формате с фиксированной запятой (обычно это целые числа), так и в формате с плавающей запятой.

Но и для формата с плавающей запятой тоже есть максимальное число. В нашем «подопытном» калькуляторе это число:

	9	9	9	9	9	е	+	9	9
--	---	---	---	---	---	---	---	---	---

То есть $99999 \cdot 10^{99}$. Самое большое по модулю отрицательное значение $-99999 \cdot 10^{99}$. Данные числа являются целыми, но именно они ограничивают представление любых чисел (целых и вещественных) в калькуляторе.

В компьютере всё организовано аналогично, но предельные значения еще больше. Это зависит от разрядности ячейки памяти, выделяемой под число, и от того, сколько разрядов выделяется под порядок и под мантиссу.

Рассмотрим пример: пусть под всё число в компьютере выделяется 8 байтов — 64 бита, из них под порядок — 2 байта, под мантиссу — 6 байтов. Тогда диапазон вещественных чисел, в переводе в десятичную систему счисления, оказывается следующим:

$$\pm(5 \cdot 10^{-324} - 1,7 \cdot 10^{308}).$$

Завершая тему, посмотрим на рис. 1.9. Смысл, заложенный в нем, такой: непрерывное, бесконечное и не ограниченное множество вещественных чисел, которое рассматривает математика, при его представлении в компьютере обращается в дискретное, конечное и ограниченное множество.



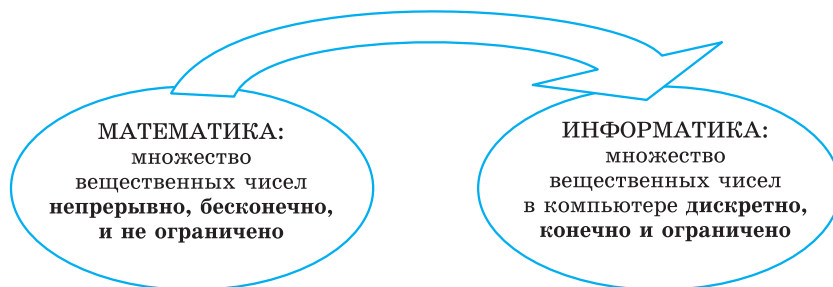


Рис. 1.9. Представление о множестве вещественных чисел в математике и в информатике



Система основных понятий

Представление чисел			
Целые числа		Вещественные числа	
<i>В математике:</i>	<i>В компьютере (информатике):</i>	<i>В математике:</i>	<i>В компьютере (информатике):</i>
- десятичное представление;	- двоичное представление;	- десятичное представление;	- двоичное представление;
- множество: дискретно, бесконечно, не ограничено	- множество: дискретно, конечно, ограничено	- множество: непрерывно, бесконечно, не ограничено	- множество: дискретно, конечно, ограничено
Представление целых чисел в компьютере		Представление вещественных чисел в компьютере	
Со знаком (положительные и отрицательные)	Без знака (положительные)	$M \times 2^P$ M — двоичная мантисса, P — двоичный целый порядок	
Диапазон: $[-2^{N-1}, 2^{N-1}-1]$	Диапазон: $[0, 2^N]$	Диапазон ограничен максимальными значениями M и P	
Формат с фиксированной запятой		Формат с плавающей запятой	

Вопросы и задания



1. Почему множество целых чисел, представимых в памяти компьютера, дискретно, конечно и ограничено?
2. Определите диапазон целых чисел, хранящихся в 1 байте памяти в двух вариантах: со знаком и без знака.
3. Получите внутреннее представление числа 127 в 8-разрядной ячейке памяти в формате со знаком.
4. Получите внутреннее представление числа –127 в 8-разрядной ячейке памяти в формате со знаком.
5. Почему множество действительных (вещественных) чисел, представимых в памяти компьютера, дискретно, конечно и ограничено?
6. На какие две части делится число в формате с плавающей запятой?



§ 6

Представление текста, изображения и звука в компьютере

В этом параграфе обсудим способы компьютерного кодирования текстовой, графической и звуковой информации. С текстовой и графической информацией конструкторы «научили» работать ЭВМ, начиная с третьего поколения (1970-е годы). А работу со звуком «освоили» лишь машины четвертого поколения, современные персональные компьютеры. С этого момента началось распространение технологии мультимедиа.

Что принципиально нового появлялось в устройстве компьютеров с освоением ими новых видов информации? Главным образом, это периферийные устройства для ввода и вывода текстов, графики, видео, звука. Процессор же и оперативная память по своим функциям изменились мало. Существенно возросло их быстродействие, объем памяти. Но как это было на первых поколениях ЭВМ, так и осталось на современных ПК — основным навыком процессора в обработке данных является умение выполнять вычисления с двоичными числами. Обработка текста, графики и звука представляет собой тоже обработку числовых данных. Если сказать еще точнее, то это *обработка целых чисел*. По этой причине компьютерные технологии называют *цифровыми технологиями*.

О том, как текст, графика и звук сводятся к целым числам, будет рассказано дальше. Предварительно отметим, что здесь мы снова встретимся с *главной формулой информатики*:

$$2^i = N.$$

Смысл входящих в нее величин здесь следующий: i — разрядность ячейки памяти (в битах), N — количество различных целых положительных чисел, которые можно записать в эту ячейку.

Текстовая информация

Принципиально важно, что текстовая информация уже дискретна — состоит из отдельных знаков. Поэтому возникает лишь технический вопрос — как разместить ее в памяти компьютера.

Напомним о байтовом принципе организации памяти компьютеров, обсуждавшемся в курсе информатики основной школы. Вернемся к рис. 1.5. Каждая клеточка на нем обозначает бит памяти. Восемь подряд расположенных битов образуют байт памяти. Байты пронумерованы. Порядковый номер байта определяет его адрес в памяти компьютера. Именно по адресам процессор обращается к данным, читая или записывая их в память (рис. 1.10).



Рис. 1.10. Байтовая организация памяти

Модель представления текста в памяти весьма проста. За каждой буквой алфавита, цифрой, знаком препинания и иным общепринятым при записи текста символом закрепляется определенный двоичный код, длина которого фиксирована. В популярных системах кодировки (Windows-1251, KOI8 и др.) каждый символ заменяется на *8-разрядное целое положительное двоичное число*; оно хранится в одном байте памяти. Это число является порядковым номером символа в кодовой таблице. Согласно главной формуле информатики, определяем, что размер алфавита, который можно закодировать, равен: $2^8 = 256$. Этого количества вполне достаточно для размещения двух алфавитов естественных языков (английского и русского) и всех необходимых дополнительных символов.

Поскольку в мире много языков и много алфавитов, постепенно совершается переход на международную систему кодировки Unicode, в которой используются многобайтовые коды. Например, если код символа занимает 2 байта, то с его помощью можно закодировать $2^{16} = 65\,536$ различных символов.

При работе с электронной почтой почтовая программа иногда нас спрашивает, не хотим ли мы прибегнуть к кодировке Unicode для пересылаемых сообщений. Таким способом можно избежать проблемы несоответствия кодировок, из-за которой иногда не удается прочитать русский текст.

Текстовый документ, хранящийся в памяти компьютера, состоит не только из кодов символьного алфавита. В нем также содержатся коды, управляющие форматами текста при его отображении на мониторе или на печати: тип и размер шрифта, положение строк, поля и отступы и пр. Кроме того, текстовые процессоры (например, Microsoft Word) позволяют включать в документ и редактировать такие «нелинейные» объекты, как таблицы, оглавления, ссылки и гиперссылки, историю вносимых изменений и т. д. Всё это также представляется в виде последовательности байтовых кодов.

Графическая информация

Из курса информатики 7–9 классов вы знакомы с общими принципами компьютерной графики, с графическими технологиями. Здесь мы немного подробнее, чем это делалось раньше, рассмотрим способы представления графических изображений в памяти компьютера.

Принцип дискретности компьютерных данных справедлив и для графики. Здесь можно говорить о *дискретном представлении изображения (рисунка, фотографии, видеок кадров) и дискретности цвета*.

Дискретное представление изображения. Изображение на экране монитора дискретно. Оно составляется из отдельных точек, которые называются пикселями (*picture elements* — элементы рисунка). Это связано с техническими особенностями устройства экрана, независимо от его физической реализации, будь то монитор на электронно-лучевой трубке, жидкокристаллический или плазменный. Эти «точки» столь близки друг другу, что глаз не различает промежутков между ними, поэтому изображение воспринимается как непрерывное, сплошное. Если выводимое из

компьютера изображение формируется на бумаге (принтером или плоттером), то линии на нем также выглядят непрерывными. Однако в основе всё равно лежит печать близких друг к другу точек.

В зависимости от того, на какое графическое разрешение экрана настроена операционная система компьютера, на нем могут размещаться изображения, имеющие размер 800×600 , 1024×768 и более пикселей. Такая прямоугольная матрица пикселей на экране компьютера называется **растром**.

Качество изображения зависит не только от размера раstra, но и от размера экрана монитора, который обычно характеризуется длиной диагонали. Существует параметр **разрешения** экрана. Этот параметр измеряется в точках на дюйм (по-английски *dots per inch* — dpi). У монитора с диагональю 15 дюймов размер изображения на экране составляет примерно 28×21 см². Зная, что в одном дюйме 25,4 мм, можно рассчитать, что при работе монитора в режиме 800×600 пикселей разрешение экранного изображения равно 72 dpi.

При печати на бумаге разрешение должно быть намного выше. Полиграфическая печать полноцветного изображения требует разрешения 200–300 dpi. Стандартный фотоснимок размером 10×15 см² должен содержать примерно 1000×1500 пикселей.

Дискретное представление цвета. Восстановим ваши знания о кодировании цвета, полученные из курса информатики основной школы. Основное правило звучит так: любой цвет точки на экране компьютера получается путем смешивания трех базовых цветов: красного, зеленого, синего. Этот принцип называется **цветовой моделью RGB** (Red, Green, Blue).

Двоичный код цвета определяет, в каком соотношении находятся интенсивности трех базовых цветов. Если все они смешиваются в одинаковых долях, то в итоге получается белый цвет. Если все три компоненты «выключены», то цвет пикселя — черный. Все остальные цвета лежат между белым и черным.

Дискретность цвета состоит в том, что *интенсивности базовых цветов могут принимать конечное число дискретных значений*.

Пусть, например, размер кода цвета пикселя равен 8 битам — 1 байту. Между базовыми цветами они могут быть распределены так:

К	К	З	З	З	С	С	С
---	---	---	---	---	---	---	---

2 бита — под красный цвет, 3 бита — под зеленый и 3 бита — под синий.

Интенсивность красного цвета может принимать $2^2 = 4$ значения, интенсивности зеленого и синего цветов — по $2^3 = 8$ значений. Полное число цветов, которые кодируются 8-разрядными кодами, равно: $4 \cdot 8 \cdot 8 = 256 = 2^8$. Снова работает главная формула информатики.

Из описанного правила, в частности, следует:

Красный		Зеленый			Синий			
0	0	0	0	0	0	0	0	— код черного цвета
1	1	1	1	1	1	1	1	— код белого цвета
0	1	0	0	1	0	0	1	— код бледно-серого цвета
0	0	1	1	1	0	0	0	— код ярко-зеленого цвета
0	0	0	0	1	0	0	0	— код бледно-зеленого цвета

Обобщение этих частных примеров приводит к следующему правилу. Если размер кода цвета равен b битов, то количество цветов (размер палитры) вычисляется по формуле:

$$K = 2^b.$$

Величину b в компьютерной графике называют **битовой глубиной цвета**.

Еще один пример. Битовая глубина цвета равна 24. Размер палитры будет равен:

$$K = 2^{24} = 16\,777\,216.$$

В компьютерной графике используются разные цветовые модели для изображения *на экране, получаемого путем излучения света*, и изображения *на бумаге, формируемого с помощью отражения света*. Первую модель мы уже рассмотрели — это модель RGB. Вторая модель носит название CMYK.

Цвет, который мы видим на листе бумаги, — это отражение белого (солнечного) света. Нанесенная на бумагу краска поглощает часть палитры, составляющей белый цвет, а другую часть отражает. Таким образом, нужный цвет на бумаге получают

путем «вычитания» из белого света «ненужных красок». Поэтому в цветной полиграфии действует не правило сложения цветов (как на экране компьютера), а правило вычитания. Мы не будем углубляться в механизм такого способа цветообразования. Расшифруем лишь аббревиатуру CMYK: Cyan — голубой, Magenta — пурпурный, Yellow — желтый, black — черный.

Растровая и векторная графика

О двух технологиях компьютерной графики — растровой и векторной — вы знаете из курса информатики основной школы.

В **растровой графике** графическая информация — это совокупность данных о цвете каждого пикселя на экране. Это то, о чем говорилось выше. В **векторной графике** графическая информация — это данные, математически описывающие графические примитивы, составляющие рисунок: прямые, дуги, прямоугольники, овалы и пр. Положение и форма графических примитивов представляются в системе экранных координат.

Растровую графику (редакторы растрового типа) применяют при разработке электронных (мультимедийных) и полиграфических изданий. Растровые иллюстрации редко создают вручную с помощью компьютерных программ. Чаще для этой цели используют сканированные иллюстрации, подготовленные художником на бумаге, или фотографии. Для ввода растровых изображений в компьютер применяются цифровые фото- и видеокамеры. Большинство графических редакторов растрового типа в большей мере ориентированы не на создание изображений, а на их обработку.

Достоинство растровой графики — эффективное представление изображений фотографического качества. Основной недостаток растрового способа представления изображения — большой объем занимаемой памяти. Для его сокращения приходится применять различные способы сжатия данных. Другой недостаток растровых изображений связан с искажением изображения при его масштабировании. Поскольку изображение состоит из фиксированного числа точек, увеличение изображения приводит к тому, что эти точки становятся крупнее. Увеличение размера точек раstra визуально искажает иллюстрацию и делает ее грубой.

Векторные графические редакторы предназначены в первую очередь для создания иллюстраций и в меньшей степени для их обработки.

Достоинства векторной графики — сравнительно небольшой объем памяти, занимаемой векторными файлами, масштабирование изображения без потери качества. Однако средствами векторной графики проблематично получить высококачественное художественное изображение. Обычно средства векторной графики используют не для создания художественных композиций, а для оформительских, чертежных и проектно-конструкторских работ.

Графическая информация сохраняется в файлах на диске. Существуют разнообразные форматы графических файлов. Они делятся на растровые и векторные. Растровые графические файлы (форматы JPEG, BMP, TIFF и другие) хранят информацию о цвете каждого пикселя изображения на экране. В графических файлах векторного формата (например, WMF, CGM) содержатся описания графических примитивов, составляющих рисунок.

Следует понимать, что графические данные, помещаемые в видеопамять и выводимые на экран, имеют растровый формат вне зависимости от того, с помощью каких программных средств (растровых или векторных) они получены.

Звуковая информация

Принципы дискретизации звука («оцифровки» звука) отражены на рис. 1.11.

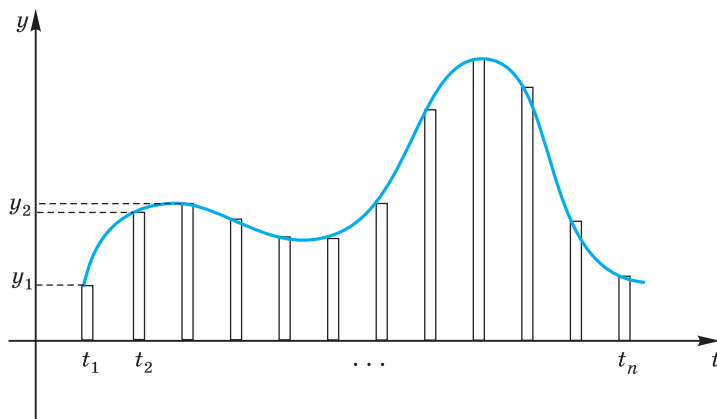


Рис. 1.11. Оцифровка звука (y — интенсивность (уровень) звукового сигнала, t — время)

Ввод звука в компьютер производится с помощью звукового устройства (микрофона, радио и др.), выход которого подключается к порту **звуковой карты**. Задача звуковой карты — с определенной частотой производить измерения уровня звукового сигнала (преобразованного в электрические колебания) и результаты измерения записывать в память компьютера. Этот процесс называют оцифровкой звука.

Промежуток времени между двумя измерениями называется периодом измерений — τ с. Обратная величина называется **частотой дискретизации** — $1/\tau$ (герц). Чем выше частота измерений, тем выше качество цифрового звука.

Результаты таких измерений представляются целыми положительными числами с конечным количеством разрядов. Вы уже знаете, что в таком случае получается дискретное конечное множество значений в ограниченном диапазоне. Размер этого диапазона зависит от разрядности ячейки — *регистра памяти звуковой карты*. Снова работает формула 2^i , где i — разрядность регистра. Число i называют также **разрядностью дискретизации**. Записанные данные сохраняются в файлах специальных звуковых форматов.

Существуют программы обработки звука — **редакторы звука**, позволяющие создавать различные музыкальные эффекты, очищать звук от шумов, согласовывать с изображениями для создания мультимедийных продуктов и т. д. С помощью специальных устройств, генерирующих звук, звуковые файлы могут преобразовываться в звуковые волны, воспринимаемые слухом человека.

При хранении оцифрованного звука приходится решать проблему уменьшения объема звуковых файлов. Для этого кроме *кодирования данных без потерь*, позволяющего осуществлять стопроцентное восстановление данных из сжатого потока, используется *кодирование данных с потерями*. Цель такого кодирования — добиться схожести звучания восстановленного сигнала с оригиналом при максимальном сжатии данных. Это достигается путем использования различных алгоритмов, сжимающих оригинальный сигнал путем выкидывания из него слабослышимых элементов. Методов сжатия, а также программ, реализующих эти методы, существует много.

Для сохранения звука без потерь используется универсальный звуковой формат файлов WAV. Наиболее известный формат «сжатого» звука (с потерями) — MP3. Он обеспечивает сжатие данных в 10 раз и более.

Система основных понятий



Дискретные модели данных					
Текст		Графика		Звук	
Таблицы кодировки		Дискретность изображения	Дискретность цвета		Дискретные измерения звукового сигнала
8-разрядные	16-разрядная	Растр — сетка пикселей	Модели цвета		Частота дискретизации
ASCII, KOI8 и др.	Unicode	$M \times N$ — размер растра, dpi — разрешение	RGB Излучаемый свет	СМЮК Отражаемый свет	$1/\text{период (герц)}$ $2^i = N$, i — разрядность, N — количество уровней измерения звука

Вопросы и задания



1. Когда компьютеры начали работать с текстом, с графикой, со звуком?
2. Что такое таблица кодировки? Какие существуют таблицы кодировки?
3. На чем основывается дискретное представление изображения?
4. Что такое модель цвета RGB?
5. Напишите 8-разрядный код ярко-синего цвета, ярко-желтого (смесь красного с зеленым), бледно-желтого.
6. Почему в полиграфии не используется модель RGB?
7. Что такое СМЮК?
8. Какое устройство в компьютере производит оцифровку вводимого звукового сигнала?
9. Как (качественно) качество цифрового звука зависит от частоты дискретизации и разрядности дискретизации?
10. Чем удобен формат MP3?

ЭОР к главе 1 на сайте ФЦИОР (<http://fcior.edu.ru>)

- Аппаратное и программное обеспечение для представления звука
- Аппаратное и программное обеспечение для представления изображения
- Единицы измерения информации
- Представление текста в различных кодировках
- Числа в памяти ЭВМ. Средства обработки числовой информации
- Числа с фиксированной и плавающей запятой
- Число и его компьютерный код

Глава 2

ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ

§ 7

Хранение информации

Из курса основной школы вам известно:

Человек хранит информацию в собственной памяти, а также в виде записей на различных внешних (по отношению к человеку) носителях: на камне, папирусе, бумаге, магнитных и оптических носителях и пр. Благодаря таким записям, информация передается не только в пространстве (от человека к человеку), но и во времени — из поколения в поколение.

Рассмотрим способы хранения информации более подробно.

Информация может храниться в различных видах: в виде записанных текстов, рисунков, схем, чертежей, фотографий, звукозаписей, кино- или видеозаписей. В каждом случае применяются свои носители.

Носитель — это материальная среда, используемая для записи и хранения информации.



Практически носителем информации может быть любой материальный объект. Информацию можно сохранять на камне, дереве, стекле, ткани, песке, теле человека и т. д. Здесь мы не станем обсуждать различные исторические и экзотические варианты носителей. Ограничимся современными средствами хранения информации, имеющими массовое применение.

Использование бумажных носителей информации

Носителем, имеющим наиболее массовое употребление, до сих пор остается бумага. Изобретенная во II веке н. э. в Китае бумага служит людям уже 19 столетий.

Для сопоставления объемов информации на разных носителях будем пользоваться единицей — байтом, считая, что один

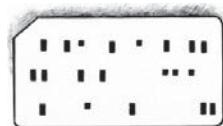
знак текста «весит» 1 байт. Нетрудно подсчитать информационный объем книги, содержащей 300 страниц с размером текста на странице примерно 2000 символов. Текст такой книги имеет объем примерно 600 000 байтов, или 586 Кб. Средняя школьная библиотека, фонд которой составляют 5000 томов, имеет информационный объем приблизительно $2861 \text{ Мб} = 2,8 \text{ Гб}$.



Что касается долговечности хранения документов, книг и прочей бумажной продукции, то она очень сильно зависит от качества бумаги, красителей, используемых при записи текста, условий хранения. Интересно, что до середины XIX века (с этого времени для производства бумаги начали использовать древесину) бумага делалась из хлопка и текстиль-

ных отходов — тряпья. Чернилами служили натуральные красители. Качество рукописных документов того времени было довольно высоким, и они могли храниться тысячи лет. С переходом на древесную основу, с распространением машинописи и средств копирования, с началом использования синтетических красителей срок хранения печатных документов снизился до 200–300 лет.

На первых компьютерах бумажные носители использовались для цифрового представления вводимых данных. Это были перфокарты: картонные карточки с отверстиями, хранящие двоичный код вводимой информации. На некоторых типах ЭВМ для тех же целей применялась перфорированная бумажная лента.



Использование магнитных носителей информации

В XIX веке была изобретена магнитная запись. Первоначально она использовалась только для сохранения звука. Самым первым носителем магнитной записи была стальная проволока диаметром до 1 мм. В начале XX столетия для этих целей использовалась также стальная катаная лента. Тогда же (в 1906 г.) был выдан



и первый патент на магнитный диск. Качественные характеристики всех этих носителей были весьма низкими. Достаточно сказать, что для производства 14-часовой магнитной записи устных докладов на Международном конгрессе в Копенгагене в 1908 г. потребовалось 2500 км, или около 100 кг проволоки.

В 20-х годах XX века появляется магнитная лента сначала на бумажной, а позднее — на синтетической (лавсановой) основе, на поверхность которой наносится тонкий слой ферромагнитного порошка. Во второй половине XX века на магнитную ленту научились записывать изображение, появляются видеокамеры, видеомэгнитофоны.

На ЭВМ первого и второго поколений магнитная лента использовалась как единственный вид сменного носителя для устройств внешней памяти. Любая компьютерная информация на любом носителе хранится в двоичном (цифровом) виде. Поэтому независимо от вида информации — текст это, или изображение, или звук — ее объем можно измерить в битах и байтах. На одну катушку с магнитной лентой, использовавшейся в лентопротяжных устройствах первых ЭВМ, помещалось приблизительно 500 Кб информации.

С начала 1960-х годов в употребление входят компьютерные магнитные диски: алюминиевые или пластмассовые диски, покрытые тонким магнитным порошковым слоем толщиной в несколько микрон. Информация на диске располагается по круговым концентрическим дорожкам, на которые она записывается и считывается в процессе вращения диска с помощью магнитных головок.

На первых ПК использовались гибкие магнитные диски (флорпи-диски) — сменные носители информации с небольшим объемом памяти — до 2 Мб. Начиная с 1980-х годов, в ПК начали использоваться встроенные в системный блок **накопители**



на жестких магнитных дисках, или НЖМД (англ. *HDD* — Hard Disk Drive). Их еще называют винчестерскими дисками.

Винчестерский диск представляет собой пакет магнитных дисков, надетых на общую ось, которая при работе компьютера находится в постоянном вращении. С каждой магнитной поверхностью пакета дисков контактирует своя магнитная головка.

Информационная емкость современных винчестерских дисков измеряется в терабайтах.

Оптические диски и флеш-память

Применение **оптического**, или **лазерного**, способа записи информации начинается в 1980-х годах. Его появление связано с изобретением квантового генератора — лазера, источника очень тонкого (толщина порядка микрона) луча высокой энергии. Луч способен выжигать на поверхности плавкого материала двоичный код данных с очень высокой плотностью. Считывание происходит в результате отражения от такой «перфорированной» поверхности лазерного луча с меньшей энергией («холодного» луча). Первоначально на ПК вошли в употребление оптические **компакт-диски** — **CD**, информационная емкость которых составляет от 190 Мб до 700 Мб.

Во второй половине 1990-х годов появились цифровые универсальные видеодиски **DVD** (Digital Versatile Disk) с большой емкостью, измеряемой в гигабайтах (до 17 Гб). Увеличение их емкости по сравнению с CD связано с использованием лазерного луча меньшего диаметра, а также двухслойной и двусторонней записи. Вспомните пример со школьной библиотекой. Весь ее книжный фонд можно разместить на одном DVD.



В настоящее время оптические диски (CD и DVD) являются наиболее надежными материальными носителями информации, записанной цифровым способом. Эти типы носителей бывают как однократно записываемыми — пригодными только для чтения, так и перезаписываемыми — пригодными для чтения и записи.

В последнее время появилось множество мобильных цифровых устройств: цифровые фото- и видеокамеры, МРЗ-плееры, карманные компьютеры, мобильные телефоны, устройства для чтения электронных книг, GPS-навигаторы и др. Все эти устройства нуждаются в переносных носителях информации. Но поскольку все мобильные устройства довольно миниатюрные, к носителям информации для них предъявляются особые требования. Они должны быть компактными, обладать низким энергопотреблением при работе, быть энергонезависимыми при хранении, иметь большую емкость, высокие скорости записи и чтения, долгий срок службы. Всем этим требованиям удовлетворяют **флеш-карты** памяти. Информационный объем флеш-карты может составлять несколько гигабайтов.

В качестве внешнего носителя для компьютера широкое распространение получили так называемые флеш-брелоки (их называют в просторечии «флешки»), выпуск которых начался в 2001 году. Большой объем информации, компактность, высокая скорость чтения/записи, удобство в использовании — основные достоинства этих устройств. Флеш-брелок подключается к USB-порту компьютера и позволяет скачивать данные со скоростью около 10 Мб в секунду.



В последние годы активно ведутся работы по созданию еще более компактных носителей информации с использованием нанотехнологий, работающих на уровне атомов и молекул вещества. В результате один компакт-диск, изготовленный по нанотехнологии, сможет заменить тысячи оптических дисков. По предположениям экспертов, приблизительно через 20 лет плотность хранения информации возрастет до такой степени, что на носителе объемом примерно с кубический сантиметр можно будет записать каждую секунду человеческой жизни.



Система основных понятий

Хранение информации							
Носители информации							
Нецифровые	Цифровые (компьютерные)						
Исторические: камень, дерево, папирус, пергамент, шелк и др.	Магнитные			Оптические		Флеш-носители	
	Ленты	Диски	Карты	CD	DVD	Флеш-карты	Флеш-брелоки
	Факторы качества носителей						
	Вместимость — плотность хранения данных, объем данных			Надежность хранения — максимальное время сохранности данных, зависимость от условий хранения			
	Наибольшей вместимостью и надежностью на сегодня обладают оптические носители CD и DVD						
Современные: бумага	Перспективные виды носителей: носители на базе нанотехнологий						



Вопросы и задания



1. Какая, с вашей точки зрения, сохраняемая информация имеет наибольшее значение для всего человечества, для отдельного человека?
2. Назовите известные вам крупные хранилища информации.
3. Можно ли человека назвать носителем информации?
4. Где и когда появилась бумага?
5. Когда была изобретена магнитная запись? Какими магнитными носителями вы пользуетесь или пользовались?
6. Какое техническое изобретение позволило создать оптические носители информации? Назовите типы оптических носителей.
7. Назовите сравнительные преимущества и недостатки магнитных и оптических носителей.
8. Что означает свойство носителя «только для чтения»?
9. Какими устройствами, в которых используются флеш-карты, вы пользуетесь? Какой у них информационный объем?
10. Какие перспективы, с точки зрения хранения информации, открывают нанотехнологии?



§ 8

Передача информации

Из курса основной школы вам известно:

- Распространение информации происходит в процессе ее передачи.
- Процесс передачи информации протекает от источника к приемнику по информационным каналам связи.

В этом параграфе более подробно будут рассмотрены технические системы передачи информации.

В § 2 уже говорилось о том, что первой в истории технической системой передачи информации стал телеграф. В 1876 году американец Александр Белл изобрел телефон. На основании открытия немецким физиком Генрихом Герцем электромагнитных волн (1886 год), А. С. Попов в России в 1895 году и почти одновременно с ним в 1896 году Г. Маркони в Италии изобрели радио. Телевидение и Интернет появились в XX веке.

Модель передачи информации К. Шеннона

Все перечисленные способы информационной связи основаны на передаче на расстояние физического (электрического или электромагнитного) сигнала и подчиняются некоторым общим законам. Исследованием этих законов занимается **теория связи**, возникшая в 1920-х годах. Математический аппарат теории связи — математическую теорию связи разработал американский ученый Клод Шеннон.

Клодом Шенноном была предложена модель процесса передачи информации по техническим каналам связи, представленная схемой на рис. 2.1.

Работу такой схемы можно пояснить на знакомом всем процессе разговора по телефону. Источником информации является говорящий человек. Кодировующим устройством — микрофон телефонной трубки, с помощью которого звуковые волны (речь) преобразуются в электрические сигналы. Каналом связи служит телефонная сеть (провода, коммутаторы телефонных узлов, через которые проходит сигнал). Декодировующим устройством является телефонная трубка (наушник) слушающего человека — приемни-



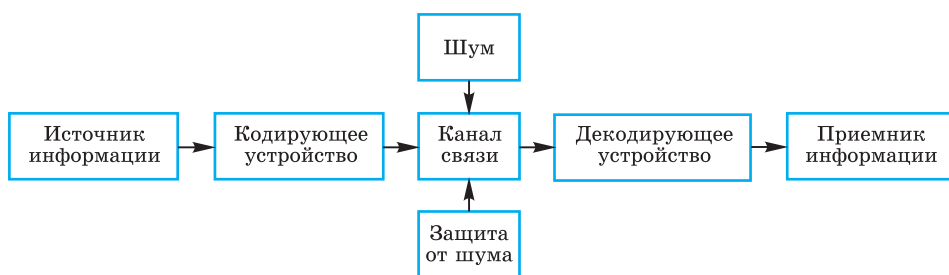


Рис. 2.1. Модель передачи информации по техническим каналам связи

ка информации. Здесь пришедший электрический сигнал превращается в звук.

В § 2 уже говорилось о кодировании на примере передачи информации через письменный документ. Кодирование там было определено как процесс представления информации в виде, удобном для ее хранения и/или передачи.

Применительно к процессу передачи информации по технической системе связи *под кодированием понимается любое преобразование информации, идущей от источника, в форму, пригодную для ее передачи по каналу связи.*

Современные компьютерные системы передачи информации — компьютерные сети, работают по тому же принципу. Есть процесс кодирования, преобразующий двоичный компьютерный код в физический сигнал того типа, который передается по каналу связи. Декодирование заключается в обратном преобразовании передаваемого сигнала в компьютерный код. Например, при использовании телефонных линий в компьютерных сетях функции кодирования/декодирования выполняет прибор, который называется *модемом*.

Пропускная способность канала и скорость передачи информации

Разработчикам технических систем передачи информации приходится решать две взаимосвязанные задачи: как обеспечить наибольшую скорость передачи информации и как уменьшить потери информации при передаче. Клод Шеннон был первым ученым, взявшимся за решение этих задач и создавшим новую для того времени науку — **теорию информации**.



Шеннон определил способ измерения количества информации, передаваемой по каналам связи. Им было введено понятие **пропускной способности канала как максимально возможной скорости передачи информации**. Эта скорость измеряется в битах в секунду (а также килобитах в секунду, мегабитах в секунду).

Пропускная способность канала связи зависит от его технической реализации. Например, в компьютерных сетях используются следующие средства связи:

- телефонные линии;
- электрическая кабельная связь;
- оптоволоконная кабельная связь;
- радиосвязь.

Пропускная способность телефонных линий — десятки и сотни Кбит/с; пропускная способность оптоволоконных линий и линий радиосвязи измеряется десятками и сотнями Мбит/с.

Скорость передачи информации связана не только с пропускной способностью канала связи. Представьте себе, что текст на русском языке, содержащий 1000 знаков, передается с использованием двоичного кодирования. В первом случае используется телеграфная 5-разрядная кодировка. Во втором случае — компьютерная 8-разрядная кодировка. Тогда длина кода сообщения в первом случае составит 5000 битов, во втором случае — 8000 битов. При передаче по одному и тому же каналу второе сообщение будет передаваться дольше в 1,6 раза ($8000/5000$). Отсюда, казалось бы, следует вывод: длину кода сообщения нужно делать минимально возможной.

Однако существует другая проблема, которая на рис. 2.1 отмечена словом «шум».

Шум, защита от шума

Термином «шум» называют разного рода помехи, искажающие передаваемый сигнал и приводящие к потере информации. Подобные помехи, прежде всего, возникают по техническим причинам, таким как плохое качество линий связи, незащищенность друг от друга различных потоков информации, передаваемых по одним и тем же каналам. Существуют и другие источники помех, имеющие физическое происхождение.

Иногда, например, беседуя по телефону, мы слышим шум, треск, мешающие понять собеседника, или на наш разговор накладывается разговор других людей.

Наличие шума приводит к потере передаваемой информации. В таких случаях необходима защита от шума. Для этого в первую очередь применяются технические способы защиты каналов связи от воздействия шумов. Такие способы бывают самыми разными, иногда простыми, иногда очень сложными. Например: использование экранированного кабеля вместо «голого» провода; применение разного рода фильтров, отделяющих полезный сигнал от шума и пр.

Шеннон разработал специальную **теорию кодирования**, дающую методы борьбы с шумом. Одна из важных идей этой теории состоит в том, что передаваемый по линии связи код должен быть *избыточным*. За счет этого потеря какой-то части информации при передаче может быть компенсирована. Например, если при разговоре по телефону вас плохо слышно, то, повторяя каждое слово дважды, вы имеете больше шансов на то, что собеседник поймет вас правильно.

В системах передачи информации используется так называемое **помехоустойчивое кодирование**, вносящее определенную избыточность.

Однако нельзя делать избыточность слишком большой. Это приведет к задержкам и удорожанию связи. Теория кодирования как раз и позволяет получить такой код, который будет оптимальным: избыточность передаваемой информации будет минимально возможной, а достоверность принятой информации — максимальной.

Большой вклад в научную теорию связи внес известный советский ученый Владимир Александрович Котельников. В 1940–1950-х годах им получены фундаментальные научные результаты по проблеме помехоустойчивости систем передачи информации.



**Владимир
Александрович
Котельников**
(1908–2005),
Россия

В современных системах цифровой связи для борьбы с потерей информации при передаче часто применяется следующий прием. Всё сообщение разбивается на порции — блоки. Для каждого блока *вычисляется контрольная сумма* (сумма двоичных цифр), которая передается вместе с данным блоком. В месте приема заново вычисляется контрольная сумма принятого блока и, если она не совпадает с первоначальной суммой, передача данного

блока повторяется. Так происходит до тех пор, пока исходная и конечная контрольные суммы не совпадут.

Система основных понятий



Передача информации в технических системах связи			
Модель К. Шеннона			
Процедура кодирования	Процесс передачи по каналу связи		Процедура декодирования
	Пропускная способность канала	Воздействие шумов на канал связи	
Защита информации от потерь при воздействии шума			
Кодирование с оптимально-избыточным кодом	Частичная потеря избыточной информации при передаче		Полное восстановление исходного сообщения

Вопросы и задания



1. Для чего нужна процедура кодирования передаваемой информации?
2. Что такое декодирование? Каким должен быть его результат?
3. Каким техническим средством связи вы чаще всего пользуетесь? Замечали ли вы при этом факты потери информации?
4. Назовите устройства кодирования и декодирования при использовании радиосвязи.
5. Что такое шум по отношению к системам передачи данных?
6. Какие существуют способы борьбы с шумом?
7. Пропускная способность канала связи 100 Мбит/с. Уровень шума пренебрежимо мал (например, оптоволоконная линия). Определите, за какое время по каналу будет передан текст, информационный объем которого составляет 100 Кб.
8. Пропускная способность канала связи 10 Мбит/с. Канал подвержен воздействию шума, поэтому избыточность кода передачи составляет 20%. Определите, за сколько времени по каналу будет передан текст, информационный объем которого составляет 100 Кб.



§ 9

Обработка информации и алгоритмы

Из курса основной школы вам известно:

Обработка информации, наряду с хранением и передачей, относится к основным видам информационных процессов.

Варианты обработки информации

Обработка информации производится каким-то субъектом или объектом (например, человеком или компьютером) в соответствии с определенными правилами. Будем его называть **исполнителем** обработки информации. Информация, которая подвергается обработке, представляется в виде **исходных данных**. На рисунке 2.2 в обобщенном виде представлен процесс обработки информации.



Рис. 2.2. Модель обработки информации

Можно привести множество примеров, иллюстрирующих схему на рис. 2.2.

Первый пример: ученик (исполнитель), решая задачу по математике, производит обработку информации. Исходные данные содержатся в условии задачи. Математические правила, описанные в учебнике, определяют последовательность вычислений. Результат — это полученный ответ.



Второй пример: перевод текста с одного языка на другой — это пример обработки информации, при которой не меняется ее содержание, но изменяется форма представления — другой язык. Перевод осуществляет переводчик по определенным правилам, в определенной последовательности.



Третий пример: работник библиотеки составляет картотеку книжного фонда. На каждую книгу заполняется карточка, на которой указываются все параметры книги: автор, название, год издания, объем и пр. Из карточек формируется каталог библиотеки, где эти карточки располагаются в строгом порядке. Например, в алфавитном каталоге карточки располагаются в алфавитном порядке фамилий авторов.



Четвертый пример: в телефонной книге вы ищете телефон нужной вам организации, например плавательного бассейна; или в том же библиотечном каталоге разыскиваете сведения о нужной вам книге. В обоих случаях исходными данными является информационный массив — телефонный справочник или каталог библиотеки, а также критерии поиска — название организации или фамилия автора и название книги.



Приведенные примеры иллюстрируют четыре различных вида обработки информации:

- 1) получение новой информации, новых сведений;
- 2) изменение формы представления информации;
- 3) систематизация, структурирование данных;
- 4) поиск информации.

Все эти виды обработки может выполнять как человек, так и компьютер. В чем состоит принципиальное различие между процессами обработки, выполняемыми человеком и машиной?

Если исполнителем обработки информации является человек, то правила обработки, по которым он действует, не всегда формальны и однозначны. Человек часто действует творчески, неформально. Даже однотипные математические задачи он может

решать разными способами. Работа журналиста, ученого, переводчика и других специалистов — это творческая работа с информацией, которая выполняется ими не по формальным правилам.

Об алгоритмах

Для обозначения формализованных правил, определяющих последовательность шагов обработки информации, в информатике используется понятие алгоритма.



Аль-Хорезми
(780–850 гг. н. э.)

Из курса информатики основной школы вы знаете, что слово «алгоритм» произошло от имени выдающегося математика средневекового Востока Мухаммеда аль-Хорезми, описавшего еще в IX веке правила выполнения вычислений с многозначными десятичными числами. Правила сложения, вычитания, умножения столбиком, деления «уголком», которым вас учили в младших классах, — это алгоритмы аль-Хорезми.

С понятием алгоритма в математике ассоциируется известный способ вычисления наибольшего общего делителя (НОД) двух натуральных чисел, который называют алгоритмом Евклида. В словесной форме его можно описать так:

1. Если числа не равны, то большее из них заменить на разность большего и меньшего из чисел.
2. Если два числа равны, то за НОД принять любое из них, иначе перейти к выполнению пункта 1.

Первоклассник, который не знает, что такое НОД, но умеет сравнивать целые числа и выполнять вычитание, сможет исполнить алгоритм. Действовать при этом он будет формально.

Такой формализованный алгоритм легко запрограммировать для современного компьютера. Мечта создать машину — автоматическое устройство, которое сможет без вмешательства человека производить расчеты, появилась очень давно. Для ее реализации требовались не только технические возможности, но и глубокое понимание сущности алгоритмов обработки информации и разработка формализованного способа представления таких алгоритмов.

Алгоритмические машины и свойства алгоритмов

В 30-х годах XX века возникает новая наука — **теория алгоритмов**. Вопрос, на который ищет ответ эта наука: для всякой ли задачи обработки информации может быть построен алгоритм решения? Но чтобы ответить на этот вопрос, надо сначала договориться об исполнителе, на которого должен быть ориентирован алгоритм.

Английский ученый Алан Тьюринг предложил модель такого исполнителя, получившую название «машина Тьюринга». По замыслу Тьюринга, его «машина» является универсальным исполнителем обработки любых символьных последовательностей в любом алфавите. Практически одновременно с Тьюрингом (1936–1937 гг.) другую модель алгоритмической машины описал Эмиль Поста. Машина Поста работает с двоичным алфавитом и несколько проще в своем «устройстве». Можно сказать, что машина Поста является частным случаем машины Тьюринга. Однако именно работа с двоичным алфавитом представляет наибольший интерес, поскольку, как вы знаете, современный компьютер тоже работает с двоичным алфавитом. Подробнее с машиной Поста вы познакомитесь в следующем параграфе.

На основании моделей Тьюринга, Поста и некоторых других ученые пришли к выводу о существовании алгоритмически неразрешимых задач.

Язык программирования алгоритмических машин представляет собой описание конечного числа простых команд, которые могут быть реализованы в автоматическом устройстве.

Совокупность всех команд языка исполнителя называется системой команд исполнителя алгоритмов — СКИ.

Алгоритм управления работой алгоритмической машины представляет собой конечную последовательность команд, посредством выполнения которой машина решает задачу обработки информации.

Алгоритм управления такой машиной должен обладать следующими свойствами:

- дискретностью (каждый шаг алгоритма выполняется отдельно от других);
- понятностью (в алгоритме используются только команды из СКИ);



Алан Тьюринг
(1912–1954),
Англия



- точностью (каждая команда определяет однозначное действие исполнителя);
- конечностью (за конечное число шагов алгоритма получается искомый результат).

Отметим разницу между понятиями «команда алгоритма» и «шаг алгоритма». Команда — это отдельная инструкция в описании алгоритма, а шаг алгоритма — это отдельное действие, которое исполнитель выполняет по команде. В циклических алгоритмах число шагов при выполнении алгоритма может быть больше, чем число команд в алгоритме, за счет повторного выполнения одних и тех же команд.



Система основных понятий

Обработка информации			
Виды обработки информации			
Получение новой информации (новых данных)	Изменение формы представления информации	Структурирование данных	Поиск данных
Исполнитель обработки			
Человек		Автомат (машина)	
Алгоритм обработки — формализованные правила, определяющие последовательность шагов обработки информации			
Алгоритмическая машина — автоматический исполнитель обработки знаковых последовательностей			
Модели алгоритмических машин в теории алгоритмов			
Машина Тьюринга		Машина Поста	
Свойства алгоритма			
Дискретность: каждый шаг алгоритма выполняется отдельно от других	Понятность: в алгоритме используются только команды из СКИ	Точность: каждая команда определяет однозначное действие исполнителя	Конечность: за конечное число шагов алгоритма получается искомый результат

Вопросы и задания



1. Приведите примеры процессов обработки информации, которые чаще всего вам приходится выполнять во время учебы. Для каждого примера определите исходные данные, результаты и правила обработки. К каким видам обработки относятся ваши примеры?
2. Если вы решаете задачу по математике или физике и при этом используете калькулятор, то какова ваша функция в этом процессе и какова функция калькулятора?
3. Используя алгоритм Евклида, найдите НОД для чисел 114 и 66. Сколько шагов алгоритма при этом вам пришлось выполнить?
4. Какие проблемы решает теория алгоритмов?
5. Почему калькулятор нельзя назвать алгоритмической машиной, а компьютер можно?
6. Придумайте минимально необходимую систему команд для кассового аппарата, который подсчитывает стоимость покупок и сумму сдачи покупателю. Опишите алгоритм управления работой такого автомата.



§ 10

Автоматическая обработка информации

В качестве примера автомата, выполняющего обработку информации, рассмотрим машину Э. Поста.

Алгоритм, по которому работает машина Поста, будем называть программой.

Договоримся о терминологии: под словом «*программа*» мы всегда будем понимать алгоритм, записанный по строгим правилам языка команд исполнителя — *на языке программирования* для данного исполнителя.

Опишем архитектуру машины Поста (рис. 2.3). Имеется бесконечная информационная лента, разделенная на позиции — клетки. В каждой клетке может либо стоять метка (некоторый знак), либо отсутствовать (пусто).



Эмиль Пост
(1897–1954),
США

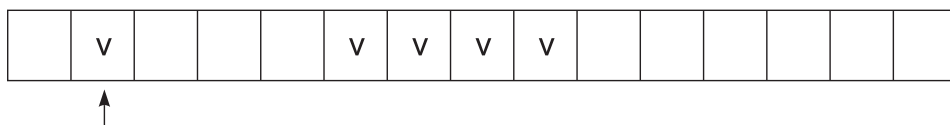


Рис. 2.3. Модель машины Поста

Вдоль ленты движется каретка — считывающее устройство. На рисунке 2.3 она обозначена стрелкой. Каретка может передвигаться шагами: один шаг — смещение на одну клетку вправо или влево. Клетку, под которой установлена каретка, будем называть текущей.

Каретка является еще и процессором машины. С ее помощью машина может:

- распознать, пустая клетка или помеченная знаком;
- стереть знак в текущей клетке;
- записать знак в пустую текущую клетку.

Если произвести замену меток на единицы, а пустых клеток — на нули, то информацию на ленте можно будет рассматривать как аналог двоичного кода телеграфного сообщения или данных в памяти компьютера. Существенное отличие каретки-процессора машины Поста от процессора компьютера состоит в том, что в компьютере возможен доступ процессора к ячейкам памяти в произвольном порядке, а в машине Поста — только последовательно.

Назначение машины Поста — производить преобразования на информационной ленте. Исходное состояние ленты можно рассматривать как исходные данные задачи, конечное состояние ленты — как результат решения задачи. Кроме того, в исходные данные входит информация о начальном положении каретки.

Теперь рассмотрим систему команд машины Поста (табл. 2.1). Запись всякой команды начинается с ее порядкового номера в программе — n . Затем следует код операции и после него — номер следующей выполняемой команды программы — m .

Рассмотрим пример программы решения задачи на машине Поста. Исходное состояние показано на рис. 2.3. Машина должна стереть знак в текущей клетке и присоединить его слева к группе знаков, расположенных справа от каретки. Программа приведена в табл. 2.2.

Таблица 2.1

Система команд машины Поста

Команда	Действие
$n \leftarrow m$	Сдвиг каретки на шаг влево и переход к выполнению команды с номером m
$n \rightarrow m$	Сдвиг каретки на шаг вправо и переход к выполнению команды с номером m
$n \vee m$	Запись метки в текущую пустую клетку и переход к выполнению команды с номером m
$n \updownarrow m$	Стирание метки в текущей клетке и переход к выполнению команды с номером m
$n !$	Остановка выполнения программы
$n ? m, k$	Переход в зависимости от содержимого текущей клетки: если текущая клетка пустая, то следующей будет выполняться команда с номером m , если непустая — команда с номером k

Таблица 2.2

Программа для машины Поста

Команда	Действие
1 $\updownarrow$ 2	Стирание метки; переход к следующей команде
2 $\rightarrow$ 3	Сдвиг вправо на один шаг
3 ? 2, 4	Если клетка пустая, то переход к команде 2, иначе — к команде 4
4 $\leftarrow$ 5	Сдвиг влево на шаг (команда выполнится, когда каретка выйдет на первый знак группы)
5 $\vee$ 6	Запись метки в пустую клетку
6 !	Остановка машины

В процессе выполнения приведенной программы многократно повторяется выполнение команд с номерами 2 и 3. Такая ситуация называется *циклом*. Напомним, что цикл относится к числу основных алгоритмических структур вместе со *следованием* и *ветвлением*.

А теперь научим машину Поста играть в интеллектуальную игру, которая называется «Игра Баше». Опишем правила игры.

Играют двое. Перед ними 21 (или 16, или 11 и т. д.) фишка. Игроки берут фишки по очереди. За один ход можно взять от 1 до 4 фишек. Проигрывает тот, кто забирает последнюю фишку.

Имеется выигрышная тактика для игрока, берущего фишки вторым. Она заключается в том, чтобы брать такое количество фишек, которое дополняет число фишек, взятых соперником на предыдущем ходе, до пяти.

Роль фишек на информационной ленте машины Поста будут выполнять метки (знаки). Машина играет с человеком. Человеку предоставляется возможность стирать метки (брать фишки) первым. Машина будет вступать в игру второй. Исходная обстановка: на ленте массив из 21 клетки содержит метки. Каретка установлена на крайней слева клетке этого массива. Стирать метки можно только подряд. Выигрышным результатом должна быть одна оставшаяся метка перед очередным ходом человека.

Еще раз напомним принцип выигрышной тактики: стирать столько меток, чтобы в сумме с метками, стертыми противником за предыдущий ход, их было пять.

Программа управления машиной Поста в игре Баше против человека приведена в табл. 2.3.

Таблица 2.3

Программа игры Баше

Команда	Действие
1 ? 2,1	Машина ждет появления пустой клетки над кареткой. После очередного хода человека машина делает свой ход. Если человек видит всего одну метку на ленте, он прекращает игру, признав свое поражение
2 → 3	Эта серия команд выведет каретку на пятую (десятую, пятнадцатую и т. д.) позицию. Какой бы ход ни сделал соперник, в ней обязательно будет стоять метка
3 → 4	
4 → 5	
5 → 6	
6 ↓ 7	Стирается метка в текущей клетке
7 ← 8	Шаг влево
8 ? 9,6	Если клетка не пустая, то возврат к команде 6
9 → 10	Каретка перемещается к первой помеченной клетке. После этого машина возвращается к команде 1 и ждет хода человека (или признания им своего поражения)
10 ? 9,1	

Действуя по данной программе и начиная стирать метки второй после человека, машина всегда будет выигрывать, если правильно задано начальное число меток, которое должно быть равно $5n + 1$, где n — любое натуральное число. В противном случае машина может проиграть.

Подведем итог. Автоматическая обработка информации возможна, если:

- 1) информация представлена в формализованном виде — в конечном алфавите некоторой знаковой системы;
- 2) реализован исполнитель, обладающий конечной системой команд, достаточной для построения алгоритмов решения определенного класса задач обработки информации;
- 3) реализовано программное управление работой исполнителя.

Машина Поста — пример автоматического исполнителя обработки информации с ограниченными возможностями. Компьютер удовлетворяет всем вышеперечисленным свойствам. Он является универсальным автоматическим исполнителем обработки информации.

Система основных понятий



Автоматическая обработка информации		
Свойства алгоритмической машины		
Дискретное, символическое представление данных (информации) в памяти машины	Достаточная конечная система команд для решения класса задач	Автоматическое исполнение процессором программы, изменяющей содержимое памяти
Алгоритмическая машина Поста		
Двоичное представление информации на бесконечной ленте памяти	– команды выставления и удаления метки; – команды смещения каретки влево и вправо; – безусловный и условный переходы; – остановка машины	Исполнение начинается с первой команды и заканчивается командой остановки. Процессор последовательно выполняет команды программы



Вопросы и задания

1. На информационной ленте машины Поста расположен массив из N меток. Каретка находится под крайней левой меткой. Какое состояние установится на ленте после выполнения следующей программы?
1 $\rightarrow$ 2
2 $\updownarrow$ 3
3 $\rightarrow$ 4
4 ? 5,2
5 $\leftarrow$ 6
6 $\vee$ 7
7 !
2. На информационной ленте на некотором расстоянии справа от каретки, стоящей под пустой клеткой, находится непрерывный массив меток. Требуется присоединить к правому концу массива одну метку.
3. На ленте расположен массив из $2n - 1$ меток. Составить программу отыскания средней метки и стирания ее.
4. На ленте расположен массив из $2n$ меток. Составить программу, по которой машина раздвинет на расстояние в одну клетку две половины данного массива.

§ 11

Информационные процессы в компьютере

Из курса основной школы вам известно:

- Компьютер (ЭВМ) — автоматическое, программно-управляемое устройство для работы с информацией.
- В состав компьютера входят устройства памяти (хранение данных и программ), процессор (обработка информации), устройства ввода/вывода (прием/передача информации).
- В 1946 году Джоном фон Нейманом были сформулированы основные принципы устройства ЭВМ, которые называют фон-неймановской архитектурой.
- Современный компьютер представляет собой единство аппаратуры (hardware) и программного обеспечения (software).

Серийное производство электронных вычислительных машин (ЭВМ) начинается в разных странах в 1950-х годах. Историю развития ЭВМ принято делить на поколения. Переход от одного поколения к другому связан со сменой элементной базы, на которой создавались машины, с изменением архитектуры ЭВМ,

с развитием основных технических характеристик (скорости вычислений, объема памяти и др.), с изменением областей применения и способов эксплуатации машин.

Под **архитектурой ЭВМ** понимаются наиболее общие принципы построения компьютера, реализующие программное управление его работой и взаимодействие основных функциональных узлов.



В основе архитектуры ЭВМ разных поколений лежат принципы Джона фон Неймана. Однако в процессе развития происходят некоторые отклонения от фон-неймановской архитектуры.

Однопроцессорная архитектура ЭВМ

Элементной базой ЭВМ **первого поколения** (1950-е годы) были электронные лампы, а ЭВМ **второго поколения** (1960-е годы) создавались на базе полупроводниковых элементов. Однако их архитектура была схожей. Она в наибольшей степени соответствовала принципам фон Неймана. В этих машинах один процессор управлял работой всех устройств: внутренней и внешней памяти, устройств ввода и вывода, как показано на рис. 2.4.

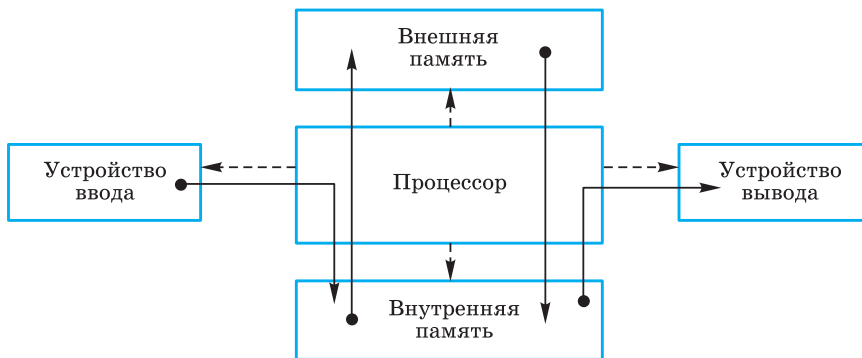


Рис. 2.4. Структура однопроцессорной ЭВМ. Сплошные стрелки — передача данных, пунктирные стрелки — управляющее воздействие

Согласно принципам фон Неймана, исполняемая программа хранится во внутренней памяти — в оперативном запоминающем устройстве (ОЗУ). Там же находятся данные, с которыми работает программа. Каждая команда программы и каждая величина (элемент данных) занимают определенные ячейки памяти, как показано на рис. 2.5.

	Внутренняя память	
	Номер ячейки	Содержимое ячейки
Программа	1	
	2	
	...	
	...	
	N	Команда STOP
Данные	$N + 1$	Величина 1
	$N + 2$	Величина 2
	...	...

Рис. 2.5. Размещение в ОЗУ программы и данных

Процессор начинает выполнение программы с первой команды и заканчивает на команде остановки, назовем ее STOP. При выполнении очередной команды процессор извлекает из памяти обрабатываемые величины и заносит их в специальные ячейки внутренней памяти процессора — регистры. Затем выполняется команда, например складываются два числа, после чего полученный результат записывается в определенную ячейку памяти. Процессор переходит к выполнению следующей команды. Исполнение программы закончится, когда процессор обратится к команде STOP.

Среди команд программы существуют команды обработки данных и команды обращения к внешним устройствам. Команды обработки данных выполняет сам процессор с помощью входящего в него арифметико-логического устройства — АЛУ, и этот процесс происходит сравнительно быстро. А команды управления внешними устройствами выполняются самими этими устройствами: устройствами ввода/вывода, внешней памятью. Время выполнения этих команд во много раз больше, чем время выполнения команд обработки данных. При однопроцессорной архитектуре ЭВМ, показанной на рис. 2.4, процессор, отдав команду внешнему устройству, ожидает завершения ее выполнения. При большом числе обращений к внешним устройствам может оказаться, что большую часть времени выполнения программы процессор «простаивает» и, следовательно, его КПД оказывается низким. Быстродействие ЭВМ с такой архитектурой находилось в пределах 10–20 тысяч операций в секунду (оп./с).

Использование периферийных процессоров

Следующим шагом в развитии архитектуры ЭВМ стал отказ от однопроцессорного устройства. Уже на последних моделях машин второго поколения, помимо *центрального процессора (ЦП)*, выполнявшего обработку данных, присутствовали *периферийные процессоры*, которые назывались каналами ввода/вывода (рис. 2.6). Их задача состояла в автономном управлении устройствами ввода/вывода и внешней памяти, что освобождало от этой работы центральный процессор. В результате КПД центрального процессора существенно возрос. Быстродействие некоторых моделей машин с такой архитектурой составляло от 1 до 3 млн оп./с.

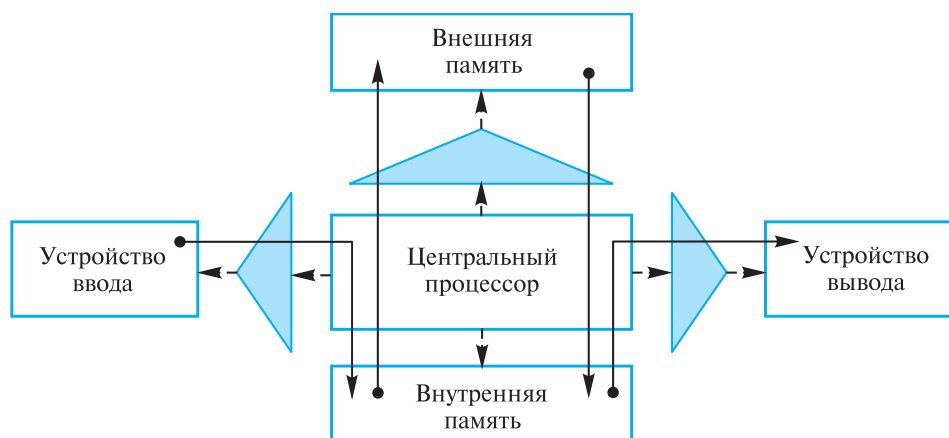


Рис. 2.6. Структура ЭВМ с одним центральным процессором и периферийными процессорами управления внешними устройствами (треугольники)

На всех моделях ЭВМ **третьего поколения**, которые создавались на базе интегральных схем (1970–80-е годы), использовалась архитектура с одним центральным процессором и периферийными процессорами внешних устройств. Такая многопроцессорная архитектура позволяла реализовать *мультипрограммный режим работы*: пока одна программа занята вводом/выводом данных, которым управляет периферийный процессор, другая программа занимает центральный процессор, выполняя вычисления. Благодаря совершенствованию элементной базы и других аппаратных средств, на некоторых моделях ЭВМ третьего поколения достигалось быстродействие до 10 млн оп./с.

Для разделения ресурсов ЭВМ между несколькими выполняемыми программами потребовалось создание специального программного обеспечения: **операционной системы (ОС)**. К разделяемым ресурсам, прежде всего, относятся время работы центрального процессора и оперативная память. Задача ОС состоит в том, чтобы разные программы, выполняемые одновременно на ЭВМ, «не мешали» друг другу и чтобы КПД центрального процессора был максимальным, иначе говоря, чтобы ЦП не «простаивал». ОС берет на себя также заботу об очередности использования несколькими программами общих внешних устройств: внешней памяти, устройств ввода/вывода.

Архитектура персонального компьютера

Персональный компьютер (ПК) — самый распространенный в наше время тип компьютера. Появление ПК связано с созданием микропроцессоров, которое началось в 1970-х годах. До недавнего времени в устройстве ПК существовал один центральный процессор и множество периферийных процессоров, управляющих внешними устройствами, которые называются контроллерами. Архитектура такого ПК изображена на рис. 2.7.

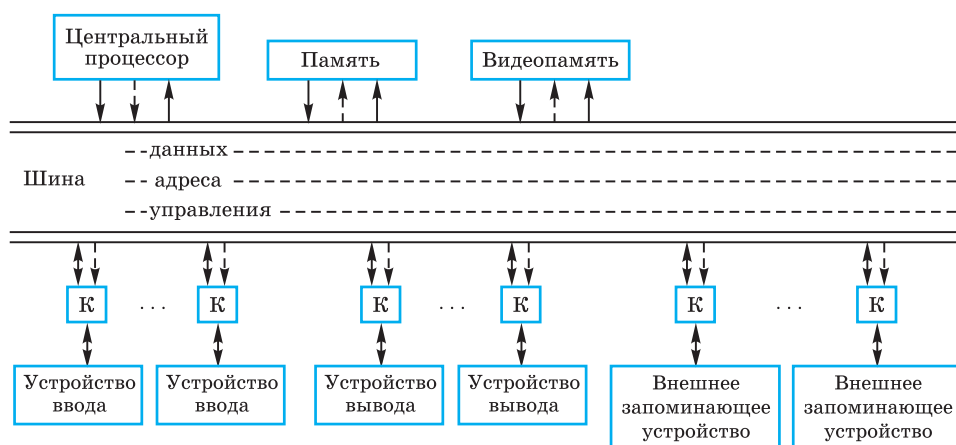


Рис. 2.7. Архитектура персонального компьютера (сплошные стрелки — направление потоков информации, пунктирные — направление управляющих сигналов, К — контроллер)

Для связи между отдельными функциональными узлами ПК используется общая информационная магистраль, которая называется **системной шиной**.

Системная шина состоит из трех частей:

- шина данных (для передачи данных);
- шина адреса (для передачи адресов устройств, которым передаются данные);
- шина управления (для передачи управляющих сигналов, синхронизирующих работу разных устройств).

Важное достоинство такой архитектуры — возможность подключения к компьютеру новых устройств или замена старых устройств на более современные. Это называется **принципом открытой архитектуры**. Для каждого типа и модели устройства используется свой контроллер, а в составе операционной системы имеется управляющая программа, которая называется драйвером устройства.

Открытая архитектура персонального компьютера — это архитектура, предусматривающая модульное построение компьютера с возможностью добавления и замены отдельных устройств.

Важное событие в совершенствовании архитектуры ПК произошло в 2005 году: был создан первый двухъядерный микропроцессор. Каждое ядро способно выполнять функции центрального процессора. Эта особенность архитектуры позволяет производить на ПК параллельную обработку данных, что существенно увеличивает его производительность. Выпускаемые в настоящее время микропроцессоры содержат до 8 ядер.

Архитектура ненејмановских вычислительных систем

Несмотря на стремительно нарастающую производительность ЭВМ, которая каждые 4–5 лет по важнейшим показателям практически удваивается, всегда есть классы задач, для которых никакой производительности не хватает. Укажем некоторые из них.

1. Математические расчеты, лежащие в основе реализации математических моделей многих процессов. Гигантские вычислительные ресурсы, которые можно реализовать очень быстро (как иногда говорят, в *масштабе реального времени*), необходимы для более надежного и долгосрочного прогноза погоды, для решения аэрокосмических задач, в том числе и оборонных, для решения многих инженерных задач и т. д.
2. Поиск информации в гигантских базах данных, в информационном пространстве Интернета.

3. Моделирование интеллекта — при всех фантастических показателях, объем оперативной памяти современных компьютеров составляет лишь малую долю объема памяти человека.

Быстродействие компьютера с одним центральным процессором имеет физическое ограничение: повышение тактовой частоты процессора ведет к повышению тепловыделения, которое не может быть неограниченным. Перспективный путь повышения производительности компьютера лежит на пути отказа от единственности главных устройств компьютера: либо процессора, либо оперативной памяти, либо шины, либо всего этого вместе. Это путь еще большего отступления от архитектуры фон Неймана.

Чтобы стало понятнее, зачем компьютеру несколько процессоров, обсудим алгоритм решения простейшей математической задачи. Есть массив из 100 чисел: $a_1, a_2, \dots, a_{100}$. Требуется найти их сумму.

Нет ничего проще! И на компьютере, и без него мы, скорее всего, поступим так: сложим первые два числа, как-то обозначим их сумму (например, S), затем прибавим к ней третье, и будем делать это еще 98 раз. Это пример последовательного вычислительного процесса. Его блок-схема приведена на рис. 2.8.

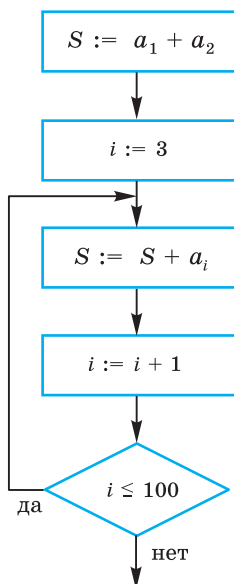


Рис. 2.8. Алгоритм решения задачи последовательного сложения массива чисел

Поскольку у человека нет второй головы, иначе эту задачу в одиночку не решить. Но представим, что мы решаем ее не в одиночку, а всем классом (25 человек). Тогда возникает возможность совсем иной последовательности действий.

1. Объединим числа в пары — по два на каждого (итого распределили 50 чисел); например, ученик № 1 берет себе a_1 и a_2 , ученик № 2 — a_3 и a_4 , и т. д.
2. Даем команду «складывай!» — и каждый складывает свои числа.
3. Даем команду «записывай!» — и каждый записывает мелом на классной доске свой результат.
4. Поскольку у нас осталось еще 50 необработанных чисел ($a_{51}, \dots, a_{100}$), повторяем пункты 1–3. После этого имеем на доске 50 чисел $b_1 = a_1 + a_2, \dots, b_{50} = a_{99} + a_{100}$ — результаты парных сложений.
5. Объединим в пары числа b_i и повторим выполнение пунктов 2–4.

Продолжаем этот процесс (2–5) до тех пор, пока не останется одно число — искомая сумма.

Первое впечатление, что всё это очень сложно, гораздо сложнее, чем алгоритм на рис. 2.8. Если бы мы захотели записать этот алгоритм в виде блок-схемы, то нам пришлось бы кроме описания порядка и объектов действий сделать то, что мы никогда при записи алгоритмов не делали, — предусмотреть синхронизацию параллельных процессов по времени. Например, выполнение команд 2 и 3 должно завершиться всеми участниками вычислений до того, как они будут продолжены (до перехода к п. 4), иначе даже при решении этой простой задачи наступит хаос.

Но сложность не есть объективная причина отвергнуть такой путь, особенно если речь идет о возможности значительного ускорения компьютерных вычислений. То, что мы предложили выше, называется на языке программистов *распараллеливанием вычислений* и вполне поддается формальному описанию. Эффект ускорения вычислений очевиден: пункт 2 в приведенном выше алгоритме ускоряет соответствующий этап работы в 25 раз!

Следующий вопрос: что надо изменить в устройстве компьютера, чтобы он смог так работать? Для реализации подобной схемы вычислений компьютеру потребуется 25 процессоров, объединенных в одну архитектуру и способных работать параллельно. Такие

многопроцессорные вычислительные комплексы — реальность сегодняшней вычислительной техники.

Вернемся, однако, к описанной выше последовательности действий — в ней еще есть источники проблем. Представим себе, что в схеме на рис. 2.7 мы дорисовали еще 24 центральных процессора, соединенных с шиной. При реализации в таком компьютере команды 3 произойдет *одновременное* обращение 25 процессоров к системной шине для пересылки результатов сложения в оперативную память. Но поскольку шина одна, числа по ней могут пересылаться только по одному! Значит, для выполнения команды 3 придется организовать очередь на передачу чисел в память. Тут же возникает вопрос: не сведет ли к нулю эта очередь все преимущества от параллельности выполнения операций на шаге 2? А если преимущества останутся, то насколько они велики? Окупятся ли расходы на 24 дополнительных процессора?

В возникшей ситуации естественен следующий шаг «изобретательской мысли»: ввод в архитектуру нескольких системных шин. А если еще подумать над возможными проблемами, то и нескольких устройств оперативной памяти.

Как видите, всё это очень непросто! Обсуждаемые изменения в устройстве компьютера приводят к «ненеймановским» архитектурам. Изобретателям таких систем приходится искать компромисс между возрастающей сложностью (и, как следствие, — стоимостью) и ускорением их работы.

Варианты реализации ненеймановских вычислительных систем

В самом общем смысле под **параллельными вычислениями** понимаются процессы обработки данных, в которых одновременно могут выполняться нескольких машинных операций. Параллельные вычисления реализуются как за счет новой архитектуры вычислительной техники, так и за счет новых технологий программирования. Такие технологии называются **параллельным программированием**.

Распределенные вычисления — способ реализации параллельных вычислений путем использования множества компьютеров, объединенных в сеть. Такие вычислительные системы еще называют **мультикомпьютерными**.

Распределенные вычисления часто реализуются с помощью **компьютерных кластеров** — нескольких компьютеров, связанных в локальную сеть и объединенных специальным программ-

ным обеспечением, реализующим параллельный вычислительный процесс. Распределенные вычисления могут производиться и с помощью **многомашинных вычислительных комплексов**, образуемых объединением нескольких отдельных компьютеров через глобальные сети.

Мультипроцессорные системы образуют единый компьютер, который относится к классу **суперкомпьютеров**. Достижение параллелизма в них происходит благодаря возможности независимой работы отдельных устройств и их дублирования: несколько процессоров, блоков оперативной памяти, шин и т. д. Мультипроцессорная система может использовать разные способы доступа к общей для всей системы памяти. Если все процессоры имеют равный (*однородный*) доступ к единой памяти, то соответствующая вычислительная система называется *векторным суперкомпьютером*.

Один из самых мощных в мире суперкомпьютеров под названием «Ломоносов» (рис. 2.9) произведен в России и работает в Московском государственном университете. Его быстродействие составляет более ста триллионов операций в секунду.



Рис. 2.9. Суперкомпьютер «Ломоносов» (МГУ им. М. В. Ломоносова)



Система основных понятий

Информационные процессы в компьютере				
Эволюция поколений ЭВМ				
	1 1950-е годы	2 1960-е годы	3 1970-е годы	4 (ПК, суперЭВМ) Начиная с 1970-х
Элемент- ная база	Электрон- ные лампы	Транзисторы	Интеграль- ные схемы (ИС) и боль- шие интег- ральные схе- мы (БИС)	БИС, СБИС (сверхболь- шие ИС), микропро- цессоры
Макси- мальное быстро- действие (оп./с)	10–20 тыс.	100 тыс. — 3 млн	10 млн	10^9 – 10^{12}
Архитек- тура	Фон-нейма- новская однопроцес- сорная	Фон-нейманов- ская однопро- цессорная. Появление периферийных процессоров	Центральный процессор + каналы вво- да/вывода. Шинная ар- хитектура	Конвейерно- векторные, матричные, многопро- цессорные, мульти- компьютер- ные системы
Ненеймановские вычислительные системы				
Ведущий принцип: отказ от последовательного выполнения операций				
Техническое решение		Программное решение		
Мультипроцессорные системы — супер- компьютеры; мульти- компьютерные системы (кластеры)		Параллельное программирование: выделение в программе нескольких одновременно вы- полняемых действий (распараллеливание)		

Вопросы и задания

1. Расскажите о смене элементной базы компьютеров, происходившей при переходе от одного поколения к другому. Как при этом менялись основные характеристики ЭВМ?
2. В чем состоял отход от архитектуры фон Неймана на ЭВМ второго и третьего поколений?
3. Что позволило реализовать мультипрограммный режим работы на ЭВМ третьего поколения?
4. Чем принципиально отличается архитектура ПК от классической архитектуры компьютеров первых поколений?
5. Какие функции выполняют контроллеры внешних устройств на ПК?
6. В чем состоит принцип открытости архитектуры ПК?
7. Какие функции выполняли первые операционные системы?
8. Для каких классов задач нужны сверхпроизводительные вычислительные системы?
9. Что такое параллельные вычисления?
10. Для примера со сложением чисел 25 учениками попробуйте проанализировать следующие ситуации: в классе всего 1 кусочек мела; в классе 5 кусочков мела; в классе 25 кусочков мела. Оцените, как от этого зависит время решения задачи (учтите еще ширину доски и время перемещения учеников по классу). Попробуйте построить модель такого процесса. Переведите эту ситуацию на язык компьютерной терминологии для многопроцессорных систем.
11. Чем отличаются мультикомпьютерные системы от мультипроцессорных? По какому принципу работают суперкомпьютеры?

**ЭОР к главе 2 на сайте ФЦИОР (<http://fcior.edu.ru>)**

- Архитектура компьютера
- Архитектура машин пятого поколения
- Внутренняя память компьютера. Внешняя память компьютера. Типы накопителей информации
- Магистраль. Передача данных внутри компьютера
- От абака до ноутбука. Поколения компьютерной техники
- Понятие алгоритма
- Принцип открытой архитектуры
- Принципы и системы передачи информации
- Принципы и системы передачи информации. Вычисление объема информации при передаче. Практическая работа

Глава 3

ПРОГРАММИРОВАНИЕ ОБРАБОТКИ ИНФОРМАЦИИ

Основы алгоритмизации и программирования вы изучали в курсе информатики основной школы.

Вы уже знакомы со следующими понятиями:

- алгоритм, исполнитель алгоритма, система команд исполнителя, свойства алгоритма;
- структуры алгоритмов: линейная ветвящаяся, циклическая;
- вспомогательный алгоритм;
- программа, языки программирования.

Вы имеете опыт программирования на языке Паскаль.

Целью изучения данного раздела является закрепление ваших навыков в алгоритмизации, углубление знаний языка программирования Паскаль и умений его практического использования для решения задач.

§ 12

Алгоритмы и величины

Этапы решения задачи на компьютере

Работа по решению любой задачи с использованием компьютера делится на следующие этапы:

- 
1. Постановка задачи.
 2. Формализация задачи.
 3. Построение алгоритма.
 4. Составление программы на языке программирования.
 5. Отладка и тестирование программы.
 6. Проведение расчетов и анализ полученных результатов.

Часто эту последовательность называют *технологической цепочкой решения задачи на компьютере*. Непосредственно к программированию в этом списке относятся пункты 3, 4, 5.

На этапе постановки задачи должно быть четко определено, *что дано и что требуется найти*. Здесь очень важно определить полный набор исходных данных, необходимый для решения задачи.

Второй этап — формализация задачи. Здесь чаще всего задача переводится на язык математических формул, уравнений, отношений. Если решение задачи требует математического описания какого-то реального объекта, явления или процесса, то формализация равносильна получению соответствующей *математической модели*.

Третий этап — построение алгоритма. Опытные программисты часто сразу пишут программы на языках программирования, не прибегая к каким-либо специальным способам описания алгоритмов (блок-схемам, псевдокодам). Однако в учебных целях полезно использовать эти средства, а затем переводить полученный алгоритм на язык программирования.

Первые три этапа — это работа без компьютера. Далее следует собственно программирование на определенном языке в определенной системе программирования. Последний (шестой) этап — это уже использование разработанной программы в практических целях. Выполнение учебных заданий на программирование обычно заканчивается пятым этапом, т. е. доказательством правильности составленной программы.

Таким образом, программист должен обладать следующими знаниями и навыками:

- уметь строить алгоритмы;
- знать языки программирования;
- уметь работать в соответствующей системе программирования.

Основой программистской грамотности является развитое алгоритмическое мышление.

Понятие алгоритма

Одним из фундаментальных понятий в информатике является понятие алгоритма. Сам термин «алгоритм» пришел из математики. Это слово происходит от «Algorithmi» — латинского написания имени Мухамеда аль-Хорезми (787–850 гг.), выдающегося математика средневекового Востока. В XII веке был осуществлен латинский перевод его математического трактата, из которого европейцы узнали о десятичной позиционной системе счисления и правилах арифметики многозначных чисел. Именно эти правила в то время называли алгоритмами. Сложение, вычитание, умноже-

ние «столбиком», деление «уголком» многозначных чисел — вот первые алгоритмы в математике. Правила алгебраических преобразований, вычисление корней уравнений также можно отнести к математическим алгоритмам.

В наше время понятие алгоритма трактуется шире. **Алгоритм** — это последовательность команд управления каким-либо исполнителем. В школьном курсе информатики с понятием алгоритма, с методами построения алгоритмов ученики впервые знакомятся на примерах учебных исполнителей: Робота, Черепашки, Чертежника и др. В нашем учебнике для 9 класса описан графический исполнитель — ГРИС. Эти исполнители ничего не вычисляют. Они создают рисунки на экране, перемещаются в лабиринтах, перетаскивают предметы с места на место. Таких исполнителей принято называть *исполнителями, работающими в обстановке*.

В разделе информатики под названием «Программирование» изучаются методы программного управления работой компьютера. Следовательно, в качестве исполнителя выступает компьютер. Он работает с величинами — различными информационными объектами: числами, символами, кодами и пр. Поэтому алгоритмы, предназначенные для управления компьютером, принято называть *алгоритмами работы с величинами*.

Данные и величины

Совокупность величин, с которыми работает компьютер, принято называть **данными**. По отношению к программе данные делятся на *исходные*, *результаты* (окончательные данные) и *промежуточные данные*, которые получаются в процессе вычислений (рис. 3.1).

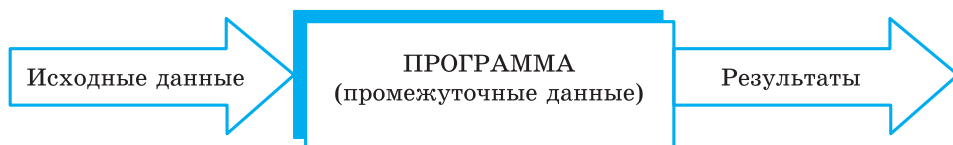


Рис. 3.1. Уровни данных относительно программы

Например, при решении квадратного уравнения: $ax^2 + bx + c = 0$ исходными данными являются коэффициенты a , b , c ; результатами — корни уравнения x_1 , x_2 ; промежуточными данными — дискриминант уравнения: $D = b^2 - 4ac$.

Для успешного освоения программирования необходимо усвоить следующее правило: *всякая величина занимает свое определенное место в памяти компьютера*. Иногда говорят — ячейку памяти. Хотя термин «ячейка», с точки зрения архитектуры современных компьютеров, несколько устарел, однако в учебных целях его удобно использовать.

У всякой величины имеются три основных свойства: **имя**, **значение** и **тип**. На уровне команд процессора величина идентифицируется адресом ячейки памяти, в которой она хранится. В алгоритмах и языках программирования величины делятся на **константы** и **переменные**. Константа — неизменная величина, и в алгоритме она представляется собственным значением, например: 15, 34.7, 'k', true. Переменные величины могут изменять свои значения в ходе выполнения программы и представляются символическими именами — идентификаторами, например: X, S2, cod15. Любая константа или переменная занимают ячейку памяти, а значение этих величин определяется двоичным кодом в этой ячейке.

Теперь о типах величин — **типах данных**. С понятием типа данных вы уже встречались, изучая в курсе информатики основной школы электронные таблицы и базы данных. Это понятие является фундаментальным для программирования.

В каждом языке программирования существует своя концепция типов данных, своя система типов. Однако в любой язык входит минимально необходимый набор основных типов данных, к которому относятся **целый, вещественный, логический и символьный** типы. С типом величины связаны три ее свойства: множество допустимых значений, множество допустимых операций, форма внутреннего представления. В таблице 3.1 представлены эти свойства основных типов данных.

Типы констант определяются по контексту (т. е. по форме записи в тексте), а типы переменных устанавливаются в описаниях переменных.

Есть еще один вариант классификации данных: классификация по структуре. Данные делятся на **простые** и **структурированные**. Для простых величин (их еще называют скалярными) справедливо утверждение: одна величина — одно значение. Для структурированных: одна величина — множество значений. К структурированным величинам относятся массивы, строки, множества и др.

Компьютер — исполнитель алгоритмов. Как известно, всякий алгоритм (программа) составляется для конкретного исполнителя, в рамках его системы команд. О каком же исполнителе идет

Таблица 3.1

Тип	Значения	Операции	Внутреннее представление
Целый	Целые положительные и отрицательные числа в некотором диапазоне. Примеры: 23, -12, 387	Арифметические операции с целыми числами: +, -, *, целочисленное деление и остаток от деления. Операции отношений (<, >, = и др.)	Формат с фиксированной запятой
Вещественный	Любые (целые и дробные) числа в некотором диапазоне. Примеры: 2.5, -0.01, 45.0, $3.6 \cdot 10^9$	Арифметические операции: +, -, *, /. Операции отношений	Формат с плавающей запятой
Логический	true (истина) false (ложь)	Логические операции: И (and), ИЛИ (or), НЕ (not). Операции отношений	1 бит: 1 — true; 0 — false
Символьный	Любые символы компьютерного алфавита. Примеры: 'a', '5', '+', '\$'	Операции отношений	Коды таблицы символьной кодировки. 1 символ — 1 байт

речь в теме «Программирование обработки информации»? Ответ очевиден: исполнителем является компьютер. Точнее говоря, исполнителем является комплекс: компьютер + система программирования (СП). Программист составляет программу на том языке, на который ориентирована СП. Схематически это изображено на рис. 3.2, где *входным языком* исполнителя является язык программирования Паскаль.



Рис. 3.2. Взаимодействие программиста с компьютером

Независимо от того, на каком языке программирования будет написана программа, алгоритм решения любой задачи на компьютере может быть составлен из команд:

- присваивания;
- ввода;
- вывода;
- обращения к вспомогательному алгоритму (подпрограмме);
- цикла;
- ветвления.

Для описания алгоритмов в дальнейшем мы будем использовать блок-схемы и учебный Алгоритмический язык, применяемый в школьном курсе информатики.

Система основных понятий



Алгоритмы и величины				
Этапы решения задачи на компьютере 1. Постановка задачи. 2. Формализация задачи. 3. Построение алгоритма. 4. Составление программы на языке программирования. 5. Отладка и тестирование программы. 6. Проведение расчетов и анализ полученных результатов				
Компьютер + система программирования = исполнитель алгоритмов обработки данных				
Система команд исполнителя	Классификация данных			
– присваивание; – ввод; – вывод; – ветвление; – цикл; – обращение к подпрограмме	По отношению к алгоритму: – исходные; – промежуточные; – итоговые (результаты)	По значениям: – константы; – переменные	По типам: – целые; – вещественные; – логические; – символьные	По структуре: – простые; – структурированные



Вопросы и задания



1. Перечислите и охарактеризуйте этапы решения задач на компьютере.
2. Дайте определение алгоритма.
3. Что такое «система команд исполнителя алгоритмов» (СКИ)?
4. Какими возможностями обладает компьютер как исполнитель алгоритмов?
5. Назовите команды, входящие в СКИ компьютера, из которых составляется любая программа обработки данных.
6. Перечислите различные варианты классификации данных.
7. Придумайте пример задачи, решаемой на компьютере, и назовите для нее исходные, промежуточные и итоговые данные.



§ 13

Структура алгоритмов

Базовые алгоритмические структуры

В 1969 году известный голландский ученый-программист Э. В. Дейкстра доказал, что *алгоритм для решения любой логической задачи можно составить только из структур следование, ветвление, цикл*. Их называют **базовыми алгоритмическими структурами**. Методика программирования, основанная на этой теореме, называется *структурным программированием*.

С базовыми алгоритмическими структурами вы познакомились, изучая информатику в 9 классе. Там же для описания структур алгоритмов были использованы два способа: блок-схемы и учебный Алгоритмический язык (АЯ). Еще раз покажем, как изображаются базовые структуры в схемах алгоритмов и как они описываются на АЯ.

Следование — это линейная последовательность действий (рис. 3.3).

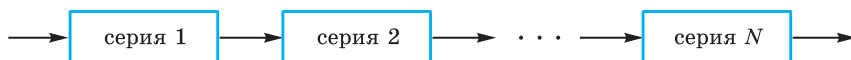


Рис. 3.3. Структура «следование»



Эдсгер В.
Дейкстра
(1930–2002)

В программе на Паскале серия — это либо один отдельный оператор, либо **составной оператор**: последовательность операторов, заключенная в операторные скобки. Например, в языке Паскаль операторными скобками являются служебные слова **Begin** и **End**.

Ветвление — алгоритмическая альтернатива. Управление передается одному из двух блоков в зависимости от истинности или ложности условия. Затем происходит выход на общее продолжение. Вот как изображается ветвление на блок-схеме и АЯ (рис. 3.4).

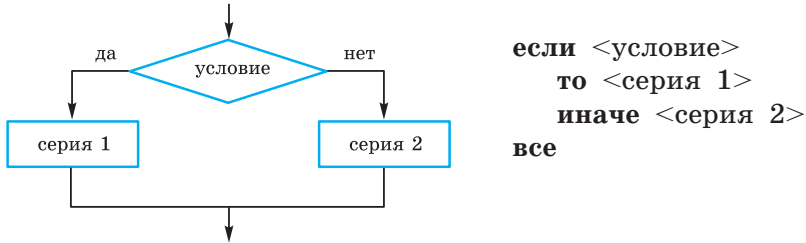


Рис. 3.4. Структура «ветвление»

Условие представляет собой утверждение, которое может быть либо истинным, либо ложным. Такое утверждение называется логическим выражением.

Неполная форма ветвления имеет место, когда на ветви «нет» пусто (рис. 3.5).

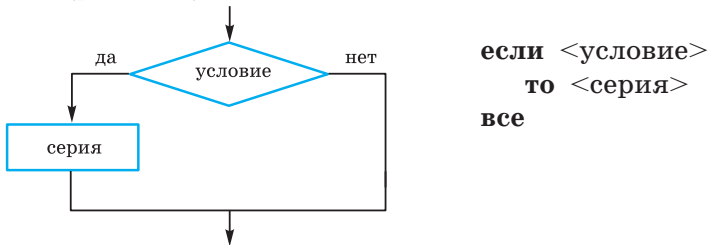


Рис. 3.5. Неполное ветвление

Цикл — повторение некоторой группы действий по условию. Различают два типа цикла. Первый — цикл с предусловием: цикл-пока (рис. 3.6).

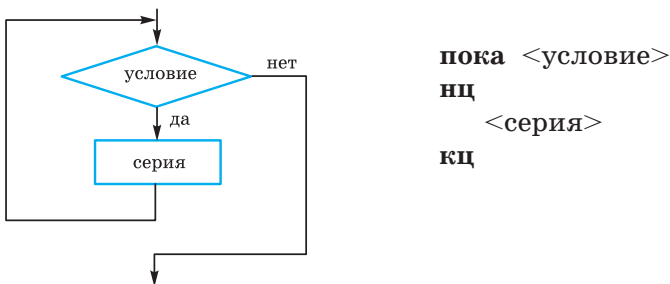


Рис. 3.6. Структура «цикл-пока»

Пока условие истинно, выполняется серия, образующая тело цикла.

Второй тип циклической структуры — **цикл с постусловием: цикл-до** (рис. 3.7).

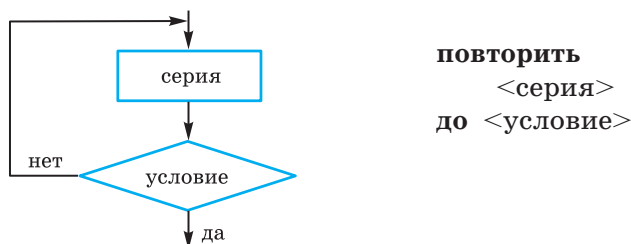


Рис. 3.7. Структура «цикл-до»

Здесь тело цикла предшествует условию цикла. Тело цикла повторяет свое выполнение, если условие ложно. Повторение прекращается, когда условие становится истинным.

Теоретически необходимым и достаточным является лишь первый тип цикла — цикл с предусловием. Любой циклический алгоритм можно построить с его помощью. Это более общий вариант цикла, чем цикл-до. В самом деле, тело цикла-до хотя бы один раз обязательно выполнится, так как проверка условия происходит после завершения его выполнения. А для цикла-пока возможен такой вариант, когда тело цикла не выполнится ни разу. Поэтому в любом языке программирования можно было бы ограничиться только циклом-пока. Однако в ряде случаев применение цикла-до оказывается более удобным, и поэтому он используется.

Иногда в литературе структурное программирование называют *программированием без GOTO* — **оператора безусловного перехода**. Действительно, при таком подходе нет места безусловному переходу. Неоправданное использование в программе оператора GOTO лишает ее структурности, а значит, всех связанных с этим положительных свойств: прозрачности и надежности алгоритма. Хотя во всех процедурных языках программирования этот оператор присутствует, однако, с точки зрения структурного подхода, его употребления следует избегать.

Комбинации базовых структур

Сложный алгоритм состоит из соединенных между собой базовых структур. Соединяться эти структуры могут двумя способами: *последовательным и вложенным*.

Если блок, составляющий тело цикла, сам является циклической структурой, то имеют место вложенные циклы. В свою очередь, внутренний цикл может иметь внутри себя еще один цикл и т. д. В связи с этим вводится представление о *глубине вложенности циклов*. Точно так же и ветвления могут быть вложенными друг в друга.

Структурный подход требует соблюдения стандарта в изображении блок-схем алгоритмов. Чертить их нужно так, как это делалось во всех приведенных примерах. Каждая базовая структура должна иметь один вход и один выход. Нестандартно изображенная блок-схема плохо читается, теряется наглядность алгоритма. Несколько примеров структурных блок-схем алгоритмов приведены на рис. 3.8 (вместо «да», «нет» здесь использованы знаки «+» и «-», У — <условие>, С — <серия>).

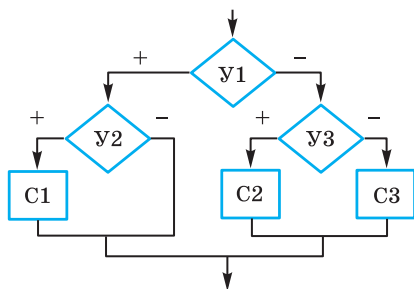
Такие блок-схемы легко читаются. Их структура хорошо воспринимается визуально. Структуре каждого алгоритма можно дать название. Приведенные блок-схемы можно охарактеризовать следующим образом (в порядке номеров).

1. Вложенные ветвления. Глубина вложенности равна единице.
2. Цикл с вложенным ветвлением.
3. Вложенные циклы-пока. Глубина вложенности — 1.
4. Ветвление с вложенной последовательностью ветвлений на положительной ветви и с вложенным циклом-пока на отрицательной ветви.
5. Следование ветвления и цикла-до.
6. Вложенные циклы. Внешний — цикл-пока, внутренний — цикл-до.

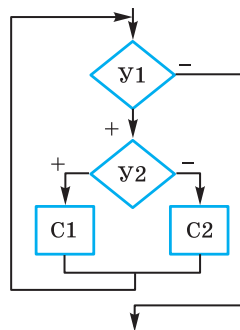
Наглядность структуре описания алгоритма на АЯ придает *структуризация внешнего вида текста*. Основной используемый для этого прием — сдвиги строк, которые должны подчиняться следующим правилам:

- конструкции одного уровня вложенности записываются на одном вертикальном уровне (начинаются с одной позиции в строке);
- вложенная конструкция записывается смещенной по строке на несколько позиций вправо относительно внешней для нее конструкции.

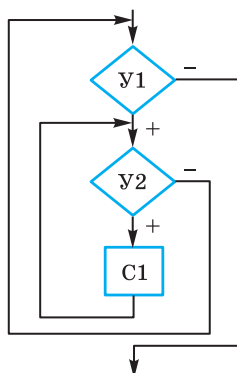
1.



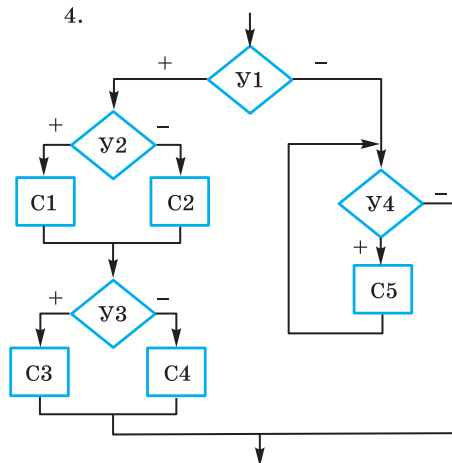
2.



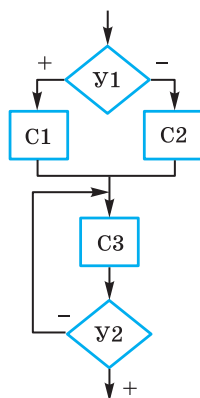
3.



4.



5.



6.

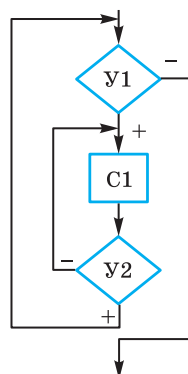


Рис. 3.8. Структурные схемы алгоритмов

Для приведенных на рис. 3.8 блок-схем структура текста на АЯ должна быть следующей:

```
1.
если <У1>
то если <У3>
    то <С1>
    все
иначе если <У2>
    то <С2>
    иначе <С3>
    все
все
```

```
3.
пока <У1>
нц
    пока <У2>
    нц
        <С1>
    кц
кц
```

```
5.
если <У1>
то <С1>
иначе <С2>
все
повторять
    <С3>
до <У2>
```

```
2.
пока <У1>
нц
    если <У2>
    то <С1>
    иначе <С2>
    все
кц
```

```
4.
если <У1>
то
    если <У2>
    то <С1>
    иначе <С2>
    все
    если <У3>
    то <С3>
    иначе <С4>
    все
иначе
    пока <У4>
    нц
        <С5>
    кц
все
```

```
6.
пока <У1>
нц
    повторять
        <С1>
    до <У2>
кц
```

Такой же способ структуризации используется и в текстах программ (например, на Паскале).

Структурное программирование — это не только форма описания алгоритма и программы, но это еще и *способ мышления программиста*. Размышляя над алгоритмом, нужно стремиться составлять его из стандартных структур. Если использовать строительную аналогию, то структурная методика построения алгоритма подобна сборке здания из стандартных секций, в отличие от складывания по кирпичику.



Система основных понятий

Структуры алгоритмов и программ		
Базовые алгоритмические структуры		
Следование	Ветвление	Цикл
Линейная последовательность действий	Выбор одной из двух серий действий с выходом на общее продолжение	Повторение серии действий по условию. Цикл с предусловием, цикл с постусловием
Комбинации базовых структур: последовательность, вложенность		
Структурный алгоритм (программа): построенный из базовых алгоритмических структур, не содержащий команды безусловного перехода (GOTO)		
Структурирование текста на Алгоритмическом языке и на языках программирования: использование сдвигов строк для вложенных конструкций		



Вопросы и задания

1. Перечислите основные базовые алгоритмические структуры и покажите способы их отображения на блок-схемах и в АЯ.
2. Какой алгоритм называется структурным?
3. Нарисуйте блок-схемы и напишите на АЯ два варианта алгоритма решения задачи: выбрать из двух числовых величин наибольшее значение. Первый вариант — с полным ветвлением, второй вариант — с неполным ветвлением.
4. Нарисуйте блок-схемы и напишите на АЯ два варианта алгоритма решения задачи: выбрать из трех числовых величин наименьшее значение. Первый вариант — с вложенными ветвлениями, второй вариант — с последовательными ветвлениями.
5. Для данного натурального числа N требуется вычислить сумму: $1 + 1/2 + 1/3 + \dots + 1/N$. Постройте блок-схемы и напишите на АЯ два варианта алгоритма: с циклом-до и с циклом-пока.
6. Какую структуру будет иметь алгоритм решения следующей задачи? Дано целое положительное число N . Если N — четное, то вычислить $N! = 1 \cdot 2 \cdot \dots \cdot N$. Если N — нечетное, то вычислить сумму: $1 + 2 + \dots + N$. Составьте блок-схему алгоритма решения и опишите его на АЯ.



§ 14

**Паскаль — язык
структурного программирования**

Программирование для ЭВМ — процесс создания программ управления работой компьютера.

Эволюция программирования

С изобретением программно управляемых вычислительных машин появилась новая профессия — программист. На ламповых ЭВМ первого поколения программисты составляли свои программы, используя непосредственно команды процессора. При этом программисту приходилось самому распределять ячейки памяти под данные и под команды программы. Нужно было знать систему команд процессора и коды всех команд. Исходные данные и команды представлялись в форме двоичного кода, т. е. непосредственно в том виде, в котором они хранились в памяти ЭВМ. Для сокращения записи программ на специальных бланках обычно использовали двоично-восьмеричный или двоично-шестнадцатеричный код. Вот пример команды программы для одного из компьютеров первого поколения:

	Адрес команды	Код операции	1-й адрес	2-й адрес	3-й адрес
Шестнадцатеричный код	28	02	C0	C4	D8
Двоичный код	0010 1000	0000 0010	1100 0000	1100 0100	1101 1000

Такая команда называется трехадресной. Код 02_{16} относится к команде сложения. 1-й и 2-й адреса — это адреса ячеек ОЗУ, в которых хранятся слагаемые, 3-й адрес — адрес ячейки, куда заносится сумма. Сама команда хранится в ячейке ОЗУ с адресом 28_{16} .

Программирование в машинных кодах представляло собой сложный процесс. По этой причине производительность работы программистов была довольно низкой. В 1950-х годах возникает направление, которое получило название «автоматизация про-

граммирования». Основная его цель — создание средств, облегчающих и ускоряющих процесс создания программы для ЭВМ. Появляются первые языки программирования.

Первыми языками программирования были машинно-ориентированные **автокоды**. Позднее за языками такого уровня закрепилось название **ассемблеры**. Первоначально ассемблером называли программу-переводчик с языка ассемблера в машинные команды. Позднее и сам язык ассемблера стали называть именем ассемблер. Программирование на ассемблере снимает с программиста заботу о распределении памяти под данные и команды программы. Программист не должен помнить внутренние коды всех команд процессора. Вот пример той же команды сложения на ассемблере (автокоде):

```
ADD a, b, c
```

Слово ADD обозначает команду «сложить», а и b — имена переменных-слагаемых, c — переменная, куда помещается результат.

Язык ассемблер называется машинно-ориентированным по той причине, что для каждой команды процессора существует свой аналог команды на ассемблере. Поскольку разные типы ЭВМ имели разные системы команд процессора, ассемблеры у них тоже отличались. Современные ассемблеры точно так же ориентированы на определенные типы процессоров. Позже появились так называемые *макроассемблеры*, в языке которых существуют макрокоманды, соответствующие сериям команд (подпрограммам) на языке процессора.

Составление программы на ассемблере проще, чем на языке команд процессора. Работу по распределению памяти под данные и команды, перевод команд ассемблера в машинные команды берет на себя специальная системная программа — **транслятор**.

Из машинной ориентированности программ на ассемблере следует, что такие программы нельзя переносить для исполнения на другие типы ЭВМ с другой системой команд процессора. Эта проблема создавала серьезные ограничения для прикладных программистов. Кроме того, само программирование на ассемблере является достаточно сложным для массового освоения, что ограничивало использование ЭВМ в прикладных областях.

Языки программирования высокого уровня. Следующим этапом развития программирования стало создание языков программирования высокого уровня — ЯПВУ. Примеры ЯПВУ: Паскаль,

Бейсик, Фортран, Си, Java и др. Все названные ЯПВУ относятся к так называемой *процедурной парадигме программирования*. Поэтому их называют *процедурными языками программирования*. Программы на таких языках представляют собой последовательности команд, описывающих действия (процедуры) компьютера по обработке информации. Существуют другие парадигмы программирования. Относящиеся к ним языки называют *декларативными языками программирования* (Пролог, Лисп и др.). Однако мы их рассматривать не будем.

Для каждого языка существует машинно-независимый стандарт. Возможность программирования на данном ЯПВУ зависит от наличия на вашем компьютере транслятора с этого языка. Трансляторы для каждого типа компьютера создают системные программисты.

Текст программы на ЯПВУ по своей форме ближе к естественным языкам (чаще всего — английскому), к языку математики. Та же команда сложения двух величин на ЯПВУ похожа на привычную форму математического равенства:

$c := a + b$ (на Паскале);
 $c = a + b$ (на Фортране, Бейсике, Си).

Освоить программирование на языке высокого уровня гораздо проще, чем на ассемблере. Поэтому с появлением ЯПВУ значительно возросло число прикладных программистов, расширилось применение ЭВМ во многих областях.

Большое количество языков программирования появилось в 1960–1970-х годах. В 1965 году в Дартмутском университете был разработан язык Бейсик. По замыслу авторов это простой, легко изучаемый язык, предназначенный для программирования несложных расчетных задач. Наибольшее распространение Бейсик получил с появлением микроЭВМ и персональных компьютеров.

История Паскаля

Язык программирования Паскаль был создан швейцарским профессором Никлаусом Виртом в 1969 году как язык для обучения студентов структурной методике программирования. Язык получил свое название в честь Блеза Паскаля, изобретателя первого вычислительного механического устройства. Позднее фирма Borland International, Inc (США) разработала систему программирования Турбо Паскаль для персональных компьютеров, которая вышла за рамки учебного применения и стала использоваться для научных и

производственных целей. В Турбо Паскаль были внесены некоторые дополнения к базовому стандарту Паскаля, описанному Н. Виртом. Со временем язык развивался. Начиная с версии 5.5, в Турбо Паскаль вводятся средства поддержки объектно-ориентированного программирования (ООП). В дальнейшем это привело к созданию Object Pascal. В начале 1990-х годов объединение элементов ООП в Паскале с визуальной технологией программирования привело к созданию системы программирования Delphi.

В настоящее время, наряду с Delphi, широко используется такая версия языка Паскаль, как PascalABC.NET, разработанная отечественными программистами и интегрированная в многоязыковую среду исполнения .NET Framework.

Структура процедурных языков программирования высокого уровня

Во всяком языке программирования определены способы организации данных и способы организаций действий над данными. Кроме того, существует понятие «элементы языка», включающее в себя множество символов (алфавит), служебных слов и других изобразительных средств языка программирования. Несмотря на разнообразие процедурных языков, их изучение происходит приблизительно по одной схеме. Это связано с общностью структуры различных процедурных языков программирования высокого уровня, которая схематически отражена на рис. 3.9.

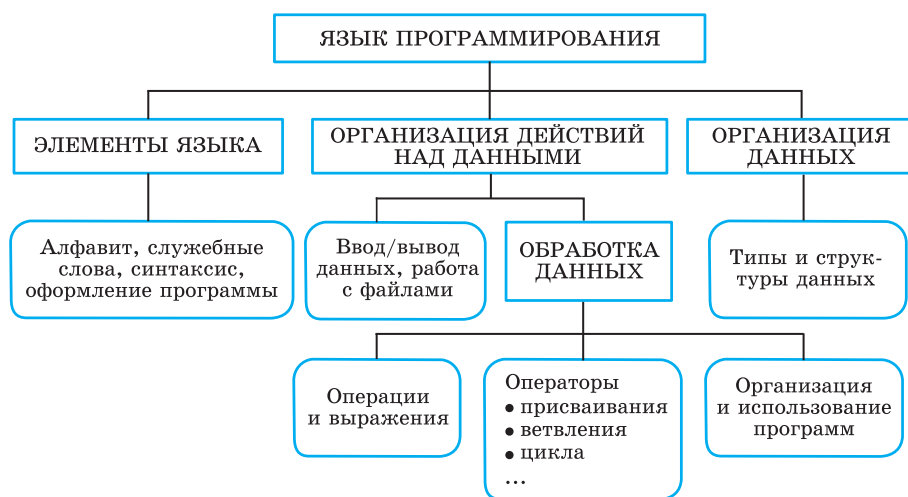


Рис. 3.9. Структура процедурного ЯПВУ



Всякий язык программирования образуют *три его основные составляющие: алфавит, синтаксис и семантика*. **Алфавит** — это множество символов, допустимых в записи текстов программ. **Синтаксис** — это правописание языковых конструкций (имен, констант, выражений, операторов и пр.). **Семантика** — это смысловое содержание языковой конструкции.

Соблюдение правил в языке программирования должно быть более строгим, чем в разговорном языке. Человеческая речь содержит значительное количество избыточной информации. Не расслышав какое-то слово, можно понять смысл фразы в целом. Слушающий или читающий человек может додумать, дополнить, исправить ошибки в воспринимаемом тексте. Компьютер же — автомат, воспринимающий всё буквально. В текстах программ нет избыточности, компьютер сам не исправит даже очевидной (с точки зрения человека) ошибки. Он может лишь указать на место, которое «не понял», и вывести замечание о предполагаемом характере ошибки. Исправить же ошибку должен программист.

Структура программы на Паскале

По определению стандартного Паскаля, программа состоит из **заголовка** программы и **тела** программы (**блока**), за которым следует *точка* — признак конца программы. В свою очередь, блок содержит **разделы описаний** (меток, констант, типов, переменных, подпрограмм) и **раздел операторов**.

```
Program <имя программы>;
Label <раздел меток>;
Const <раздел констант>;
Type <раздел типов>;
Var <раздел переменных>;
Procedure (Function) <раздел подпрограмм>;
Begin
    <раздел операторов>
End.
```

Раздел операторов имеется в любой программе и является основным. Предшествующие разделы носят характер описаний и не все обязательно присутствуют в каждой программе.

В PascalABC.NET, в отличие от базового стандарта Паскаля, возможно:

- отсутствие заголовка программы;
- разделы **Const**, **Type**, **Var**, **Label** могут следовать друг за другом в любом порядке и повторяться в разделе описаний сколько угодно раз.



Система основных понятий

Паскаль — язык структурного программирования		
Эволюция программирования		
Программирование в машинных кодах (начало 1950-х годов)	Машинно-ориентированные языки программирования: автокоды, ассемблеры (2-я половина 1950-х годов и позже)	Машинно-независимые языки программирования высокого уровня — ЯПВУ (1960-е годы и позже)
Структура процедурного ЯПВУ		
Элементы языка: алфавит, служебные слова и др.	Организация действий над данными: ввод/вывод, операции, операторы	Организация данных: типы и структуры данных
Паскаль — процедурный язык структурного программирования. Этапы развития: стандартный Паскаль (Н. Вирт), Турбо Паскаль, Object Pascal, Delphi, PascalABC.NET		
Структура программы на Паскале		
Заголовок программы	Раздел описаний	Раздел операторов



Вопросы и задания

1. В каком виде составлялись программы для первых компьютеров?
2. Чем отличались программы на автокодах (ассемблерах) от программ в машинных кодах?
3. Почему ЯПВУ являются машинно-независимыми языками программирования?
4. Что такое трансляция?
5. В какой парадигме программирования реализован язык Паскаль?
6. Что входит в структуру любого процедурного ЯПВУ?
7. Из каких основных разделов состоит программа на Паскале?

§ 15

Элементы языка Паскаль и типы данных

Алфавит. Алфавит языка состоит из множества символов, включающих в себя *буквы, цифры и специальные символы*.

Латинские буквы: от *A* до *Z* (заглавные) и от *a* до *z* (строчные).

Цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Специальные символы: + — * / = < > [] . , () : ; { } ^ @ \$ #.

Следующие комбинации специальных символов являются единичными символами (их нельзя разделять пробелами):

`:=` знак присваивания;

`<=` меньше или равно;

`>=` больше или равно;

`(* *)` ограничители комментариев (наряду с `{ }`);

`<>` не равно;

`(. .)` эквивалент `[ ]`.

Пробелы — символ пробела (код ASCII 32) и все управляющие символы кода ASCII (от 0 до 31).

Служебные слова. К спецсимволам относятся и служебные слова, смысл которых определен однозначно. Служебные слова не могут быть использованы для других целей. С точки зрения языка, они являются единичными элементами алфавита. Вот некоторые служебные слова: **Program, Var, array, If, Do, While** и др.

Идентификаторы. Идентификатором называется символическое имя определенного программного объекта. Такими объектами являются: имена констант, переменных, типов данных, процедур и функций, программ. *Идентификатор — это любая последовательность букв и цифр, начинающаяся с буквы.* К буквам приравнивается также знак подчеркивания. Длина идентификатора может быть произвольной, но значащими являются только первые 63 символа.

Комментарии. Следующие конструкции представляют собой комментарии и поэтому пропускаются компилятором:

```
{любой текст, не содержащий символ "фигурная скобка"}
(* любой текст, не содержащий символы "звездочка,
   круглая скобка"*)
//последующий текст до конца строки
```

Буквы русского алфавита употребляются только в комментариях, символьных и текстовых константах.

Концепция типов данных в Паскале

Концепция типов данных является одной из центральных в любом языке программирования. Как вы знаете, с **типом величины** связаны три ее свойства: форма внутреннего представления, множество принимаемых значений и множество допустимых операций.

Паскаль характеризуется большим разнообразием типов данных, отраженным на рис. 3.10.

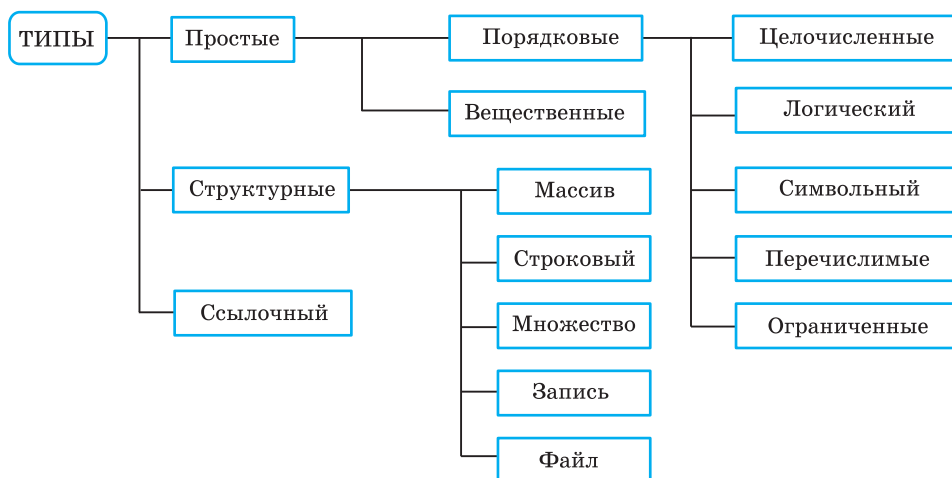


Рис. 3.10. Система типов данных Паскаля

Тип данных называется **порядковым**, если он состоит из счетного количества значений, которые можно пронумеровать. Отсюда следует, что на этом множестве значений существуют понятия «следующий» и «предыдущий».

В стандартном Паскале Вирта отсутствовал строковый тип. Он присутствует в старших версиях языка, в том числе в PascalABC.NET. Кроме того, в PascalABC.NET целые и вещественные — это группы типов.

Каждый тип имеет свой идентификатор. В таблице 3.2 представлена информация о простых типах данных, определенных в PascalABC.NET. Для вещественных типов в скобках указано количество сохраняемых значащих цифр мантиссы в десятичном представлении числа.

Таблица 3.2

Простые типы данных

Имя типа	Длина в байтах	Диапазон (множество) значений	Десятич- ных цифр в мантиссе
Целочисленные типы			
shortint	1	–128..127	
smallint	2	–32768..32767	
integer (longint)	4	–2147483648..2147483647	
int64	8	–9223372036854775808.. 9223372036854775807	
byte	1	0..255	
word	2	0..65535	
longword (cardinal)	4	0..4294967295	
unit64	8	0..18446744073709551615	
biginteger	перемен- ная	неограниченный	
Вещественные типы			
real	8	$-1.8 \cdot 10^{308} \dots 1.8 \cdot 10^{308}$	15–16
double	8	$-1.8 \cdot 10^{308} \dots 1.8 \cdot 10^{308}$	15–16
single	4	$-3.4 \cdot 10^{38} \dots 3.4 \cdot 10^{38}$	7–8
decimal	16	–79228162514264337593543950335.. 79228162514264337593543950335	28–29
Логический тип			
boolean	1	true, false	
Символьный тип			
char	2	все символы кодировки Unicode	

Типы пользователя. Одна из принципиальных возможностей Паскаля состоит в том, что пользователю разрешается определять свои типы данных. Типы пользователя всегда базируются на стандартных типах данных Паскаля. Для описания типов пользователя в Паскале существует раздел типов, начинающийся со служебного слова **Type**. К простым типам пользователя относятся перечислимый и интервальный типы данных.

Перечислимый тип задается непосредственно перечислением (списком) всех значений, которые может принимать переменная данного типа:

```
Type <имя типа> = (<список значений>)
```

Определенное таким образом имя типа затем используется для описания переменных. Например:

```
Type Gaz = (C, O, N, F);  
      Metal = (Fe, Co, Na, Cu, Zn);  
Var G1, G2, G3: Gaz;  
      Met1, Met2: Metal;  
      Day: (Sun, Mon, Tue, Wed, Thu, Fri, Sat);
```

Здесь *Gaz* и *Metal* — имена перечислимых типов, которые ставятся в соответствие переменным *G1*, *G2*, *G3* и *Met1*, *Met2*. Переменной *Day* назначается перечислимый тип, которому не присвоено имени.

Значения, входящие в перечислимый тип, являются *константами*. Действия над ними подчиняются правилам, применимым к константам. Каждое значение в перечислимом типе занимает в памяти 2 байта, поэтому число значений этого типа не должно превышать 65 535.

Перечислимый тип — упорядоченное множество. Его элементы пронумерованы, начиная от 0 в порядке следования в описании.

В программе, в которой присутствует данное выше описание переменной *Day*, возможен такой фрагмент:

```
If Day=Sun Then Write('Ура! Сегодня выходной!');
```

Ограниченный тип задается как упорядоченное ограниченное подмножество некоторого порядкового типа:

```
<константа 1>..<константа 2>
```

Порядковый номер первой константы не должен превышать номера второй константы в соответствующем базовом типе.

При исполнении программы автоматически контролируется принадлежность значений переменной ограниченного типа уста-

новленному диапазону. При выходе из диапазона исполнение программы прерывается.

Пример

```
Type Numbers = 1..31;
      Alf = 'A'..'Z';
Var   Data: Numbers;
      Bukva: Alf;
```

Структурные типы. Особенностью Паскаля является то, что в нем структуры данных рассматриваются как типы — структурные типы данных. Одна величина *простого типа* представляет собой одно значение: целое число, вещественное число, символ и пр. Одна величина *структурного типа* представляет собой совокупность множества значений; примеры — числовой массив, символьная строка и пр.

Автор Паскаля Вирт придавал большое значение разнообразию типов данных в языке программирования. В своей книге «Алгоритмы и структуры данных» он подчеркивает зависимость алгоритма решения задачи от способа организации данных в программе. Удачно выбранный способ организации данных упрощает алгоритм решения задачи.

Система основных понятий



Элементы языка и типы данных в Паскале	
Состав программы на Паскале	
Элементы языка	Комментарии
Алфавит, служебные слова, идентификаторы	Поясняющие тексты, которые пропускаются компилятором
Типы данных	
Стандартные (целочисленные, вещественные, логический, символьный)	Определяемые в программе: перечислимые, ограниченные
Простые типы	Структурные типы
Одна величина — одно значение	Одна величина — множество значений (массивы, строки, записи, множества, файлы)



Вопросы и задания

1. Входят ли в алфавит Паскаля русские буквы? Для чего их можно использовать?
2. Что такое идентификатор? Каковы правила его задания?
3. Чем различаются типы данных из группы целочисленных типов?
4. Чем различаются типы данных из группы вещественных типов?
5. В чем разница между простыми и структурными типами?
6. Что такое перечислимый и ограниченный типы данных?

§ 16

Операции, функции, выражения

Арифметические операции

К числовым типам данных относятся группы вещественных и целочисленных типов. К ним применимы арифметические операции и операции отношений.

Операции над данными бывают унарными (применимые к одному операнду) и бинарными (применимые к двум операндам).

Унарная арифметическая операция в Паскале одна. Это операция изменения знака. Ее формат:

–<величина>

Бинарные арифметические операции стандартного Паскаля описаны в табл. 3.3. В ней символ «I» обозначает целые типы, символ «R» — вещественные типы.

Таблица 3.3

Бинарные операции Паскаля

Знак	Выражение	Типы операндов	Тип результата	Операция
+	A+B	R, R I, I I, R; R, I	R I R	Сложение
–	A–B	R, R I, I I, R; R, I	R I R	Вычитание
*	A*B	R, R I, I I, R; R, I	R I R	Умножение

Окончание таблицы 3.3

Знак	Выражение	Типы операндов	Тип результата	Операция
/	A/B	R, R I, I I, R; R, I	R R R	Вещественное деление
div	A div B	I, I	I	Целочисленное деление
mod	A mod B	I, I	I	Остаток от целочисленного деления

Стандартные функции и процедуры

В Паскале существует большое количество стандартных функций и процедур, к которым программист может обращаться в своих программах. Наиболее часто используются математические функции, например: `sqrt(x)` — квадратный корень, `abs(x)` — абсолютная величина, `sin(x)` и др. Часто используемые стандартные процедуры: `Read(...)` — процедура ввода, `Write(...)` — процедура вывода данных.

Стандартные функции и процедуры являются внешними подпрограммами по отношению к вызывающей их программе. Они объединены в **модули**, которые подключаются к основной программе и становятся доступными для использования. Наиболее часто используемые подпрограммы объединены в модуль под названием `SYSTEM`. Этот модуль подключается к программе автоматически.

Таблица 3.4 содержит описания стандартных математических функций Паскаля.

Для подключения других модулей необходимо в начале программы (после заголовка) записать строку:

```
Uses <имя модуля>
```

Для управления символьным выводом на экран используется стандартный модуль `CRT`. К программе он подключается командой:

```
Uses CRT
```

В дальнейшем из этого модуля мы будем использовать *процедуру очистки экрана* для символьного вывода, обращение к которой производится оператором `ClrScr`.

Таблица 3.4

Стандартные математические функции Паскаля

Обращение	Тип аргумента	Тип результата	Функция
Pi	—	R	Число $\pi = 3,1415926536E+00$
abs(x)	I, R	I, R	Модуль аргумента
arctan(x)	I, R	R	Арктангенс (в радианах)
cos(x)	I, R	R	Косинус (в радианах)
exp(x)	I, R	R	e^x — экспонента
frac(x)	I, R	R	Дробная часть x
int(x)	I, R	R	Целая часть x
ln(x)	I, R	R	Натуральный логарифм
random	—	R	Псевдослучайное число в интервале $[0, 1)$
random(x)	I	I	Псевдослучайное число в интервале $[0, x)$
round(x)	R	I	Округление до ближайшего целого
sin(x)	I, R	R	Синус (в радианах)
sqr(x)	I, R	I, R	Квадрат x
sqrt(x)	I, R	R	Квадратный корень
trunc(x)	R	I	Ближайшее целое, не превышающее x по модулю

Арифметические выражения

Арифметическое выражение задает порядок выполнения действий над числовыми величинами. Арифметические выражения содержат числовые константы и переменные, арифметические операции, функции, круглые скобки. Одна константа или одна переменная — простейшая форма арифметического выражения.

Например, рассмотрим математическое выражение:

$$\frac{2a + \sqrt{0,5\sin(x + y)}}{0,2c - \ln(x - y)}.$$

На Паскале оно выглядит так:

```
(2*A+Sqrt(0.5*sin(X+Y)))/(0.2*C-ln(X-Y))
```

Для того чтобы правильно записывать арифметические выражения, нужно соблюдать следующие правила.

1. Все символы пишутся в строчку на одном уровне. Проставляются все знаки операций (нельзя пропускать знак *).
2. Не допускаются два следующих подряд знака операций. (Нельзя: $A+-B$; можно: $A+(-B)$.)
3. Операции с более высоким приоритетом выполняются раньше операций с меньшим приоритетом. Порядок убывания приоритетов:
 - вычисление функции;
 - унарная операция смены знака (-);
 - *, /, div, mod;
 - +, -.
4. Несколько записанных подряд операций одинакового приоритета выполняются последовательно слева направо.
5. Часть выражения, заключенная в скобки, вычисляется в первую очередь. (Например, в выражении $(A+B) * (C-D)$ умножение производится после сложения и вычитания.)

Не следует записывать выражения, не имеющие математического смысла, например деление на нуль, логарифм отрицательного числа и т. п.

Пример. Цифрами сверху указан порядок выполнения операций:

```

1  7  4  5  3      6  2   12 11 10  8   9
(1+y) * (2*x+sqrt(y)-(x+y)) / (y+1/(sqrt(x)-4))

```

Данное арифметическое выражение (на Паскале) соответствует следующему математическому выражению:

$$(1+y) \frac{2x + \sqrt{y} - (x+y)}{y + \frac{1}{x^2 - 4}}$$

В Паскале нет операции или стандартной функции возведения числа в произвольную степень. Для вычисления x^y рекомендуется поступать следующим образом:

- а) если y — целое положительное значение, то его степень вычисляется через умножение; например $x^3 \rightarrow x * x * x$; большие степени следует вычислять умножением в цикле;
- б) если y — целое отрицательное число, то степень вычисляется так: $x^y = (1/x)^{|y|}$; а при $y = 0$: $x^0 = 1$;
- в) если y — вещественное значение, не равное нулю, то используется следующая математическая формула: $x^y = e^{y \ln x}$. На Паскале получим арифметическое выражение:

`exp (y*ln (x) )`

Очевидно, что в этом случае не допускается нулевое или отрицательное значение x . Для целого y такого ограничения нет.

Пример:

$$\sqrt[3]{a+1} = (a+1)^{\frac{1}{3}}.$$

На Паскале это выражение выглядит так:

`exp (1/3*ln (a+1) )`

Выражение имеет целочисленный тип, если в результате его вычисления получается величина целочисленного типа. Выражение имеет вещественный тип, если результатом его вычисления является вещественная величина.



Система основных понятий

Операции, функции, выражения
Арифметические операции применяются к числовым величинам; нет операции возведения в степень
Тип результата операции зависит от типа операндов (табл. 3.3)
Возведение в степень (x^y): при целом y реализуется через умножение, при вещественном y реализуется так: <code>exp (y*ln (x))</code>
Стандартные функции и процедуры описаны в модулях, подключаемых к программе. Модуль SYSTEM подключается по умолчанию
Арифметическое выражение: языковая конструкция, определяющая порядок вычисления числовой величины в соответствии с математическим выражением. Типом выражения называется тип результата его вычисления

Вопросы и задания



1. Для следующих математических выражений запишите соответствующие арифметические выражения на Паскале:

а) $a + bx + cyz$; б) $[(ax - b)x + c]x - d$; в) $\frac{a + b}{c} + \frac{c}{ab}$;

г) $\frac{x + y}{a_1} \cdot \frac{a_2}{x - y}$; д) $10^4\alpha + 3\frac{1}{5}\beta$;

е) $\left(1 + \frac{x}{2!} + \frac{y}{3!}\right) / \left(1 + \frac{2}{3 + xy}\right)$.

2. Запишите математические выражения, соответствующие следующим выражениям на Паскале:

а) $(p+q) / (r+s) - p * q / (r * s)$;

б) $1E3 + beta / (x - gamma * delta)$;

в) $a / b * (c + d) - (a - b) / b / c + 1E-8$.

3. Для следующих математических выражений запишите соответствующие арифметические выражения на Паскале:

а) $(1 + x)^2$; б) $\sqrt{1 + x^2}$; в) $\cos^2 x^2$; г) $\log_2 \frac{x}{5}$;

д) $\arcsin x$; е) $\frac{e^x + e^{-x}}{2}$; ж) $x^{\sqrt{2}}$; з) $\sqrt[3]{1 + x}$;

и) $\sqrt{x^8 + 8x}$; к) $\frac{xyz - 3,3|x + \sqrt[4]{y}|}{10^7 + \ln 4!}$; л) $\frac{\beta + \sin^2 \pi^4}{\cos 2 + |\operatorname{ctg} y|}$.

4. Вычислите значения выражений:

а) $\operatorname{trunc}(6.9)$

е) $\operatorname{round}(6.2)$

б) $\operatorname{trunc}(6.2)$

ж) $20 \bmod 6$

в) $20 \operatorname{div} 6$

з) $2 \bmod 5$

г) $2 \operatorname{div} 5$

и) $3 * 7 \operatorname{div} 2 \bmod 7 / 3 - \operatorname{trunc}(\sin(1))$

д) $\operatorname{round}(6.9)$

5. Определите типы выражений:

а) $1 + 0.0$

в) $\operatorname{sqr}(4)$

д) $\sin(0)$

б) $20 / 4$

г) $\operatorname{sqr}(16)$

е) $\operatorname{trunc}(-3.14)$

§ 17

Оператор присваивания, ввод и вывод данных

Присваивание — это действие, в результате которого переменная величина получает определенное значение. В программе на Паскале существуют три способа присваивания значения переменной:

- 1) оператор присваивания;
- 2) оператор ввода;
- 3) передача значения через параметры подпрограммы.

Оператор присваивания имеет следующий формат:

`<переменная>:=<выражение>`

Например:

- 1) `x:=2*a+sqrt(b);`
- 2) `b:=(x>y) and (k<>0).`

Сначала вычисляется выражение, затем полученное значение присваивается переменной.

В первом примере приведен *арифметический оператор присваивания*. Здесь *x* — переменная вещественного типа.

Во втором примере — *логический оператор присваивания*. Здесь *b* — переменная типа `Boolean`.

Типы переменной и выражения должны совпадать. Из этого правила есть одно исключение: переменной вещественного типа можно присваивать значение целочисленного выражения. В таком случае значение целого числа преобразуется к формату с плавающей запятой и присвоится вещественной переменной.

Ввод и вывод данных

Под вводом понимается передача данных с внешнего устройства компьютера в оперативную память. При выводе данные передаются из оперативной памяти на внешнее устройство (рис. 3.11).

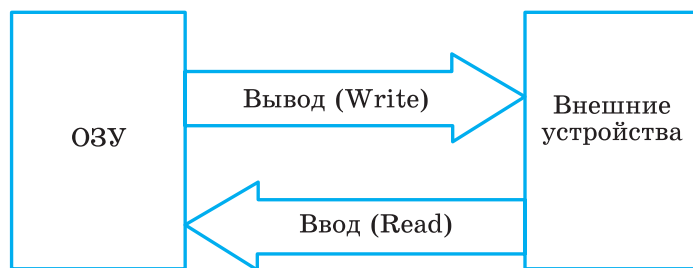


Рис. 3.11. Ввод и вывод

Операция ввода называется чтением и выполняется с помощью оператора `Read`. Вывод называется записью, и для его выполнения используется оператор `Write`.

К внешним устройствам относятся устройства ввода и вывода (клавиатура, монитор, принтер и др.) и устройства внешней памяти (магнитные и оптические диски, флеш-память и др.). Данные на внешних устройствах организованы в файлы.

Для внешних запоминающих устройств (ВЗУ) **файл** — это *поименованная область памяти* этого устройства. В файлы на ВЗУ можно записывать данные по команде `Write` и можно читать данные из файлов по команде `Read`. На одном устройстве ВЗУ может храниться множество файлов одновременно. Правила именования файлов на ВЗУ определяются операционной системой. Имена для файлов, создаваемых пользователем, задает сам пользователь.

Устройства ввода с клавиатуры и вывода на экран монитора являются однофайловыми устройствами. Считается, что с клавиатурой связан один системный файл с именем `INPUT`. Поэтому ввод с клавиатуры равнозначен чтению из файла `INPUT`. С монитором связан системный файл, который называется `OUTPUT`. Вывод на экран — это запись данных в файл `OUTPUT`.

Ввод с клавиатуры производится путем обращения к стандартной процедуре `Read` в следующем формате:

```
Read(<список ввода>)
```

Чтение происходит из системного файла `INPUT`, всегда доступного для любой программы. Элементами списка ввода могут быть переменные символьного типа, числовых типов и строковые переменные.

Например:

```
Read(a, b, c, d)
```

При выполнении этого оператора происходит прерывание исполнения программы, после чего пользователь должен набрать на клавиатуре значения переменных *a*, *b*, *c*, *d*, отделяя их друг от друга пробелами. При этом вводимые значения высвечиваются на экране. В конце нажимается клавиша Enter. Значения следует вводить в строгом соответствии с синтаксисом Паскаля.

Пример:

```
Var T: Real; J: Integer; K: Char;  
Begin  
    Read(T, J, K);
```

Набираем на клавиатуре:

```
253.98 100 G [Enter]
```

Если в программе имеется несколько подряд идущих операторов Read, то данные для них можно вводить последовательно (на экране отражаются в одной строке) и лишь в конце ввода нужно нажать клавишу Enter.

Пример:

```
Var A, B: Integer;  
    C, D: Real;  
Begin  
    Read(A, B); Read(C, D);
```

Набираем на клавиатуре и видим на экране:

```
18758 34 2.62E-02 1.54E+01 [Enter]
```

Другой вариант оператора ввода с клавиатуры имеет вид:

```
ReadLn(<список ввода>)
```

Здесь слово «ReadLn» означает *read line* — «читать строку». Нажатие клавиши Enter в процессе ввода вырабатывает признак «конец строки», и данные при выполнении следующего оператора ввода будут отражаться на экране с начала новой строки. Если в предыдущем примере заменить операторы Read на ReadLn:

```
ReadLn(A, B); ReadLn(C, D);
```

то ввод значений будет происходить из двух строк, отраженных на экране:

```
18758 34 [Enter]
2.62E-02 1.54E+01 [Enter]
```

Вывод на экран производится по оператору обращения к стандартной процедуре:

```
Write(<список вывода>)
```

Здесь элементами списка вывода могут быть выражения различных типов (в частности, константы и переменные).

Например: `Write('Сумма ', A, '+', B, '=', A+B)`

Если, например, $A = 5$ и $B = 7$, то на экране получим:

```
Сумма 5+7=12
```

При выводе на экран нескольких значений в строку они не отделяются друг от друга пробелами. Программист сам должен позаботиться о таком разделении. В приведенном примере предусмотрен пробел после слова «Сумма».

Второй вариант процедуры вывода на экран:

```
WriteLn(<список вывода>)
```

Write line — «писать строку». Его действие отличается от оператора `Write` тем, что после вывода последнего в списке значения происходит перевод курсора к началу следующей строки. Оператор `WriteLn`, записанный без параметров, вызывает перевод строки.

В списке вывода могут присутствовать указатели форматов вывода (форматы). Формат определяет представление выводимого значения на экране. Формат отделяется от соответствующего ему элемента двоеточием. Если указатель формата отсутствует, то машина выводит значение по определенному правилу, предусмотренному по умолчанию.

Линейная программа. Следование — простейшая алгоритмическая структура. Программа, реализующая следование, называется линейной программой. В линейной программе могут присутствовать только операторы присваивания, ввода, вывода и обращения к процедурам. Заметим, что операторы `Read` и `Write` являются обращениями к стандартным процедурам Паскаля.

Одним из обязательных условий хорошего стиля программирования является организация диалога между компьютером и пользователем. Такое диалоговое взаимодействие называется *интерактивным интерфейсом*.

Пример 1. Составим линейную программу, по которой в диалоге будут вводиться два целых числа и вычисляться их произведение.

```
Program Multiply;  
Var A, B, AB: integer;  
Begin  
    Write('A= '); ReadLn(A);  
    Write('B= '); ReadLn(B);  
    AB:=A*B;  
    Write(A, '*', B, '=', AB)  
End.
```

Тестирование этой программы отразится на экране следующим образом.

```
A= 13  
B= 28  
13*28=364
```

Числа 13 и 28 вводятся пользователем с клавиатуры, всё остальное автоматически выводится по программе.

Пример 2. Дано натуральное трехзначное число. Требуется вычислить сумму его цифр. Например, если дано число 325, то в результате должно получиться: $3 + 2 + 5 = 10$.

Сначала составим программу, а потом ее прокомментируем.

```
Program SumCifr;  
Var X, Sum: integer;  
Begin  
    Write('Введите трехзначное число: '); ReadLn(X);  
    Sum:=0;  
    Sum:=Sum + X mod 10;  
    X:=X div 10;  
    Sum:=Sum + X mod 10;  
    X:=X div 10;  
    Sum:=Sum + X;  
    Write('Сумма цифр = ', Sum)  
End.
```

В этой программе использованы две операции целочисленной арифметики: `div` — целочисленное деление и `mod` — остаток от целочисленного деления (см. табл. 3.3). Остаток от деления на 10 (`mod`) выделяет младшую цифру числа, а целочисленное деление на 10 (`div`) отбрасывает младшую цифру.

Чтобы лучше понять работу программы, выполним ее трассировку. В курсе 9 класса вам уже приходилось строить трассировочные таблицы. Для программы `SumCifr` таблица будет выглядеть следующим образом:

№	Команда	<i>X</i>	<i>Sum</i>
1	<code>readln(X)</code>	325	–
2	<code>Sum:=0</code>		0
3	<code>Sum:=Sum + X mod 10</code>		5
4	<code>X:=X div 10</code>	32	
5	<code>Sum:=Sum + X mod 10</code>		7
6	<code>X:=X div 10</code>	3	
7	<code>Sum:=Sum + X</code>		10
8	<code>Write('Сумма цифр = ', Sum)</code>		10

Выполнение программы на компьютере приводит к такому же результату.

Заметим, что эту задачу можно решить с помощью всего одного оператора присваивания:

```
Sum:=X mod 10 + X div 10 mod 10 + X div 100
```

Проверьте самостоятельно.



Система основных понятий

Присваивание, ввод, вывод
Оператор присваивания: <code><переменная>:=<выражение></code> Тип переменной и выражения должны совпадать. Исключение: вещественной переменной можно присваивать значение целого выражения
Ввод — передача данных с внешнего устройства в ОЗУ
Ввод с клавиатуры: <code>Read(<список ввода>)</code> или <code>ReadLn(<список ввода>)</code>
Вывод — передача данных из ОЗУ на внешнее устройство
Вывод на экран: <code>Write(<список вывода>)</code> или <code>WriteLn(<список вывода>)</code>
Линейная программа состоит из операторов ввода, вывода, присваивания и обращения к процедурам



Вопросы и задания

1. Назовите последовательность действий при выполнении оператора присваивания.
2. Сформулируйте правило соответствия типов для оператора присваивания. Какое существует исключение из этого правила?
3. Если y — вещественная переменная, а n — целая, то какие из следующих операторов присваивания правильные, а какие — нет?

а) $y:=n+1$	д) $y:=n \text{ div } 2$
б) $n:=y-1$	е) $y:=y \text{ div } 2$
в) $n:=4.0$	ж) $n:=n/2$
г) $y:=\text{trunc}(y)$	з) $n:=\text{sqr}(\text{sqr}(n))$
4. Напишите линейную программу, в результате выполнения которой целочисленные переменные x и y обменяются значениями. При этом нельзя использовать дополнительные переменные. Найдя такой алгоритм, определите, в чем его недостаток по сравнению с методом обмена через третью переменную. Можно ли его применять для вещественных чисел?
5. Напишите оператор присваивания, в результате выполнения которого целочисленной переменной h присвоится значение цифры, стоящей в разряде сотен в записи положительного целого числа k (например, если $k = 28\,796$, то $h = 7$).



6. Напишите линейную программу, в результате выполнения которой в целочисленной переменной S получится перевернутое целое четырехзначное число k . Например: если $k = 1357$, то $S = 7531$.
7. Напишите линейную программу перевода любого целого четырехзначного двоичного числа в десятичную систему счисления. Например, дано число в двоичной системе счисления: 1101_2 . Перевод в десятичную систему выполняется так: $1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 = 13$.



§ 18

Логические величины, операции, выражения

С элементами математической логики вы уже встречались в курсе информатики основной школы, изучая способы записи запросов к базе данных и условной функции ЕСЛИ в электронных таблицах, основы алгоритмизации и программирования. Повторим основные понятия логики с целью дальнейшего углубления ваших знаний в использовании ее для программирования.



К числу основных понятий логики относятся: *высказывание, логическая величина, логические операции, логические выражения и формулы*.



Высказывание (суждение) — это повествовательное предложение, в котором что-либо утверждается или отрицается. По поводу любого высказывания можно сказать, истинно оно или ложно.

Например, высказывание «На улице идет дождь» будет истинным или ложным в зависимости от состояния погоды в данный момент. Истинность высказывания «Значение A больше, чем B », записанного в форме неравенства: $A > B$, будет зависеть от значений переменных A и B .

Логические величины — понятия, выражаемые словами: ИСТИНА, ЛОЖЬ (true, false). Следовательно, *истинность высказываний выражается через логические величины*.

Логическая константа: ИСТИНА или ЛОЖЬ.

Логическая переменная: символически обозначенная логическая величина. Следовательно, если известно, что A , B , X , Y

и др. — переменные логические величины, то это означает, что они могут принимать значения только **ИСТИНА** или **ЛОЖЬ**.

Логическое выражение — простое или сложное высказывание. Сложное высказывание строится из простых с помощью логических операций (связок).

Логические операции

Конъюнкция (логическое умножение). В русском языке она выражается союзом **И**. В математической логике используются знаки $\&$ или $\wedge$. Конъюнкция — двухместная операция; записывается в виде: $A \& B$. Значением такого выражения будет **ЛОЖЬ**, если значение хотя бы одного из операндов ложно.

Дизъюнкция (логическое сложение). В русском языке этой связке соответствует союз **ИЛИ**. В математической логике она обозначается знаком $\vee$. Дизъюнкция — двухместная операция; записывается в виде: $A \vee B$. Значением такого выражения будет **ИСТИНА**, если значение хотя бы одного из операндов истинно.

Отрицание. В русском языке этой связке соответствует частица **НЕ** (в некоторых высказываниях применяется оборот «неверно, что ...»). Отрицание — унарная (одноместная) операция; записывается в виде: $\neg A$ или $\bar{A}$.

Правила выполнения рассмотренных логических операций отражены в следующей таблице, которая называется **таблицей истинности** логических операций (здесь **И** означает «истина», **Л** — «ложь»):

A	B	$\neg A$	$A \& B$	$A \vee B$
И	И	Л	И	И
И	Л	Л	Л	И
Л	И	И	Л	И
Л	Л	И	Л	Л

Логическая формула — формула, содержащая лишь логические величины и знаки логических операций. Результатом вычисления логической формулы является **ИСТИНА** или **ЛОЖЬ**.

Последовательность выполнения операций в логических формулах определяется старшинством операций. В порядке убывания старшинства логические операции расположены так: *отрицание*, *конъюнкция*, *дизъюнкция*. Кроме того, на порядок выполнения операций влияют скобки, которые можно использовать в логических формулах.

Например: $(A \& B) \vee (\neg A \& B) \vee (\neg A \& \neg B)$.

Пример. Вычислить значение логической формулы:

$$\neg X \& Y \vee X \& Z,$$

если логические переменные имеют следующие значения: $X = \text{ЛОЖЬ}$, $Y = \text{ИСТИНА}$, $Z = \text{ИСТИНА}$.

Решение. Отметим цифрами сверху порядок выполнения операций в формуле:

$$\begin{array}{cccc} 1 & 2 & 4 & 3 \\ \neg X \& Y \vee X \& Z. \end{array}$$

Используя таблицу истинности, вычислим формулу по шагам:

- 1) $\neg \text{ЛОЖЬ} = \text{ИСТИНА}$;
- 2) $\text{ИСТИНА} \& \text{ИСТИНА} = \text{ИСТИНА}$;
- 3) $\text{ЛОЖЬ} \& \text{ИСТИНА} = \text{ЛОЖЬ}$;
- 4) $\text{ИСТИНА} \vee \text{ЛОЖЬ} = \text{ИСТИНА}$.

Ответ: ИСТИНА.

Логические функции на области числовых значений

Алгебра чисел пересекается с алгеброй логики в тех случаях, когда приходится проверять принадлежность значений алгебраических выражений некоторому множеству. Например, принадлежность значения числовой переменной X множеству положительных чисел выражается через *высказывание*: « X больше нуля». Символически это записывается так: $X > 0$. В алгебре такое выражение называют неравенством. В логике — отношением.

Отношение $X > 0$ может быть истинным или ложным. Если X — положительная величина, то оно истинно, если отрицательная, то ложно. В общем виде отношение имеет следующую структуру:

<выражение 1> <знак отношения> <выражение 2>

Здесь выражения 1 и 2 — некоторые математические выражения, принимающие числовые значения. В частном случае

выражение может представлять собой одну константу или одну переменную величину. Знаки отношений могут быть следующими:

$=$	— равно;
$\neq$	— не равно;
$\geq$	— больше или равно;
$\leq$	— меньше или равно;
$>$	— больше;
$<$	— меньше.

Например:

$$x = 5; \quad a + b \neq x - 1; \quad b^2 - 4ac \geq 0; \quad \sin(x) < x/2.$$

Итак, отношение — это простое высказывание, а значит, логическая величина. Оно может быть как постоянной: $5 > 0$ — всегда ИСТИНА, $3 \neq 6 : 2$ — всегда ЛОЖЬ; так и переменной: $a < b$, $x + 1 = c - d$. Если в отношение входят переменные числовые величины, то и значение отношения будет логической переменной.

Отношение можно рассматривать как **логическую функцию** от числовых аргументов. Например: $F(x) = (x > 0)$ или $P(x, y) = (x < y)$. Аргументы определены на бесконечном множестве действительных чисел, а значения функции — на множестве, состоящем из двух логических величин: ИСТИНА, ЛОЖЬ.

Логические функции от числовых аргументов еще называют термином **предикат**. В алгоритмах предикаты играют роль условий, по которым строятся ветвления и циклы. Предикаты могут быть как простыми логическими функциями, не содержащими логических операций, так и сложными, содержащими логические операции.

Пример 1. Записать предикат (логическую функцию) от двух вещественных аргументов X и Y , который будет принимать значение ИСТИНА, если точка на координатной плоскости с координатами X и Y лежит внутри единичной окружности с центром в начале координат (рис. 3.12).

Из геометрических соображений понятно, что для всех точек, лежащих

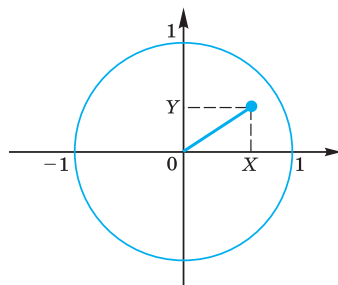


Рис. 3.12

внутри единичной окружности, будет истинным значение следующей логической функции:

$$F(X, Y) = (X^2 + Y^2 < 1).$$

Для значений координат точек, лежащих на окружности и вне ее, значение функции F будет ложным.

Пример 2. Записать предикат, который будет принимать значение ИСТИНА, если точка на координатной плоскости с координатами X и Y лежит внутри кольца с центром в начале координат и радиусами $R1$ и $R2$.

Поскольку значения $R1$ и $R2$ — переменные величины, искомая логическая функция будет иметь четыре аргумента: X , Y , $R1$, $R2$. Возможны две ситуации:

- 1) $R1^2 < X^2 + Y^2 < R2^2$ и $R1 < R2$: $R1$ — внутренний радиус, $R2$ — внешний радиус;
- 2) $R2^2 < X^2 + Y^2 < R1^2$ и $R2 < R1$: $R2$ — внутренний радиус, $R1$ — внешний радиус.

Объединив дизъюнкцией оба этих утверждения и записав их по правилам алгебры логики, получим следующую логическую функцию:

$$F(X, Y, R1, R2) = (((X^2 + Y^2) > R1^2) \& ((X^2 + Y^2) < R2^2) \& R1 < R2) \vee \\ \vee (((X^2 + Y^2) > R2^2) \& ((X^2 + Y^2) < R1^2) \& R2 < R1).$$

Пример 3. Записать предикат, который будет принимать значение ИСТИНА, если точка на координатной плоскости с координатами X и Y лежит внутри фигуры, ограниченной жирными линиями на рис. 3.13.

Фигура ограничена тремя границами, описываемыми уравнениями:

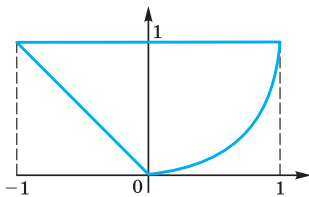


Рис. 3.13

$Y = -X$ — левая граница, линейная функция;

$Y = 1$ — верхняя граница, константа;

$Y = X^2$ — правая граница, парабола.

Рассматриваемая область есть пересечение трех полуплоскостей, описываемых неравенствами:

$$\begin{cases} Y > -X; \\ Y < 1; \\ Y > X^2. \end{cases}$$

Во внутренних точках все эти три отношения являются одновременно истинными. Поэтому искомый предикат имеет вид:

$$F(X, Y) = (Y > -X) \& (Y < 1) \& (Y > X^2).$$

Логические выражения на Паскале

Уже говорилось о том, что в Паскале имеется логический тип данных.

Логические константы: `true` (истина), `false` (ложь).

Логические переменные: описываются с типом `Boolean`.

Операции отношения: осуществляют сравнение двух операндов и определяют, истинно или ложно соответствующее отношение между ними. Знаки операций отношения: `=` (равно), `<>` (не равно), `>` (больше), `<` (меньше), `>=` (больше или равно), `<=` (меньше или равно).

Логические операции: `not` — отрицание, `and` — логическое умножение (конъюнкция), `or` — логическое сложение (дизъюнкция), `xor` — исключающее ИЛИ. Таблица истинности для этих операций (Т — `true`; F — `false`):

A	B	<code>not A</code>	<code>A and B</code>	<code>A or B</code>	<code>A xor B</code>
T	T	F	T	T	F
T	F	F	F	T	T
F	T	T	F	T	T
F	F	T	F	F	F

Логическое выражение может состоять из логических констант и переменных, отношений, логических операций. Логическое выражение принимает значение `true` или `false`.

Например, логическая формула $\neg X \& Y \vee X \& Z$ на Паскале запишется в виде следующего логического выражения:

not X **and** Y **or** X **and** Z,

где X, Y, Z — переменные типа Boolean.

Логические операции располагаются в следующем порядке по убыванию старшинства (приоритета): 1) **not**, 2) **and**, 3) **or**, **xor**. Операции отношения имеют самый низкий приоритет. Поэтому если операндами логической операции являются отношения, то их следует заключать в круглые скобки. Например, математическому неравенству $1 \leq X \leq 50$ соответствует следующее логическое выражение:

(1<=X) **and** (X<=50)

Логическая функция `odd(x)` принимает значение `true`, если значение целочисленного аргумента `x` является нечетным, иначе — `false`.

Для правильной записи сложного логического выражения (предиката) нужно учитывать относительные приоритеты арифметических, логических операций и операций отношений, поскольку все они могут присутствовать в логическом выражении. По убыванию приоритета операции располагаются в следующем порядке.

1. Арифметические операции:

– (минус унарный)

*, /

+, –

2. Логические операции:

not

and

or, **xor**

3. Операции отношения:

=, <>, >, <, >=, <=

Еще раз обратите внимание, что в логическом выражении, соответствующем предикату из примера 3:

(Y>-X) **and** (Y<1) **and** (Y>X*X),

операции отношения заключены в скобки, поскольку они младше логических операций, а выполняться должны раньше.



Система основных понятий

Логические величины, операции, выражения			
Базовые понятия логики			
Высказывание (суждение)	Логическая величина	Логическое выражение	Основные логические операции
Некоторое утверждение, которое может быть либо истинным, либо ложным	Принимает одно из двух значений: ИСТИНА , ЛОЖЬ	Простое или сложное высказывание, содержащее логические связки (операции)	- конъюнкция (логическое умножение) — И ; - дизъюнкция (логическое сложение) — ИЛИ ; - инверсия (отрицание) — НЕ
Логические величины в Паскале: true — ИСТИНА , false — ЛОЖЬ			
Логические операции в языке Паскаль			
not — отрицание	and — конъюнкция	or — дизъюнкция	xor — исключающее ИЛИ
Изменяет логическое значение операнда на противоположное	Равно true , если оба операнда true	Равно false , если оба операнда false	Равно true , если операнды имеют разные значения
Старшинство операций в логическом выражении на Паскале			
1. Арифметические операции: 1) — (минус унарный) 2) *, / 3) +, -	2. Логические операции: 1) not 2) and 3) or , xor	3. Операции отношения: =, <>, >, <, >=, <=	

Вопросы и задания



1. Какого типа величина получается при вычислении отношения (неравенства) между числами?
2. Что такое предикат? Приведите примеры.
3. Запишите на языке алгебры логики логические функции, которые будут принимать значение ИСТИНА, если справедливы следующие утверждения, и ЛОЖЬ — в противном случае:
 - а) все числа X, Y, Z равны между собой;
 - б) из чисел X, Y, Z только два равны между собой;
 - в) каждое из чисел X, Y, Z положительно;
 - г) только одно из чисел X, Y, Z положительно;
 - д) значения чисел X, Y, Z упорядочены по возрастанию.
4. Все формулы, полученные при решении предыдущей задачи, запишите в виде логических выражений на Паскале.
5. Постройте таблицу истинности для логической формулы:

$$\neg X \& Y \vee X \& Z.$$

Пояснение: в таблице истинности должны быть вычислены значения формулы для всех вариантов значений логических переменных: X, Y, Z . Следовательно, таблица будет содержать $2^3 = 8$ строк и 4 столбца: значения X, Y, Z и результат. В таблицу можно добавить дополнительные столбцы, содержащие результаты промежуточных операций.

6. Вычислите значения следующих логических выражений, записанных на Паскале:

а) $K \bmod 7 = K \operatorname{div} 5 - 1$	при $K = 15$;
б) $\text{odd}(\text{trunc}(10 * P))$	при $P = 0,182$;
в) $\text{not odd}(n)$	при $n = 0$;
г) $t \text{ and } (P \bmod 3 = 0)$	при $t = \text{true}, P = 10101$;
д) $(x * y < 0) \text{ and } (y > x)$	при $x = 2, y = 1$;
е) $a \text{ or not } b$	при $a = \text{false}, b = \text{true}$.

Пояснения: $\text{odd}(x)$ — логическая функция определения четности аргумента, равна true , если x — нечетное, и равна false , если x — четное; $\text{trunc}(x)$ — целочисленная функция от вещественного аргумента, возвращающая ближайшее целое число, не превышающее x по модулю.

§ 19

Программирование ветвлений

В § 13 был показан способ отображения ветвления (полного и неполного) на блок-схеме и учебном Алгоритмическом языке. Алгоритмическая структура ветвления программируется в Паскале с помощью **условного оператора If**. В 9 классе вы познакомились с этим оператором. Вспомним его формат.

Полное ветвление:

```
If <логическое выражение>  
Then <оператор 1>  
Else <оператор 2>
```

Неполное ветвление:

```
If <логическое выражение>  
Then <оператор>
```

То, что в алгоритмах называется условием, в Паскале является *логическим выражением*, которое вычисляется в первую очередь. Если его значение равно `true`, то будет выполняться <оператор 1> (после **Then**), если — `false`, то <оператор 2> (после **Else**) для полной формы или оператор, сразу следующий после условного, для неполной формы (без **Else**). На ветвях может быть как простой оператор, так и составной — серия операторов в операторных скобках **Begin**, **End**.

Пример 1. По длинам трех сторон треугольника a , b , c требуется вычислить его площадь.

Для решения задачи используется формула Герона

$$\sqrt{p(p-a)(p-b)(p-c)},$$

где $p = (a + b + c)/2$ — полупериметр треугольника. Исходные данные должны удовлетворять основному соотношению для сторон треугольника — длина каждой стороны должна быть меньше суммы длин двух других сторон, и длины сторон не могут быть отрицательными величинами.

Имея возможность в одном условном операторе записывать достаточно сложные логические выражения, используя логические операции, мы можем сразу «отфильтровать» все варианты неверных исходных данных.



```

Program Geron;
Var A, B, C, P, S : Real;
Begin
  WriteLn('Введите длины сторон треугольника: ');
  Write('a= '); ReadLn(A);
  Write('b= '); ReadLn(B);
  Write('c= '); ReadLn(C);
  If (A>0) and (B>0) and (C>0) and (A+B>C)
    and (B+C>A) and (A+C>B)
  Then Begin
    P := (A+B+C)/2;
    S := Sqrt(P*(P-A)*(P-B)*(P-C));
    WriteLn('Площадь=', S)
  End
  Else WriteLn('Неверные исходные данные')
End.

```

Пример 2. Требуется перевести пятибалльную оценку в ее наименование: 5 — «отлично», 4 — «хорошо», 3 — «удовлетворительно», 2 — «неудовлетворительно».

Блок-схема алгоритма приведена на рис. 3.14.

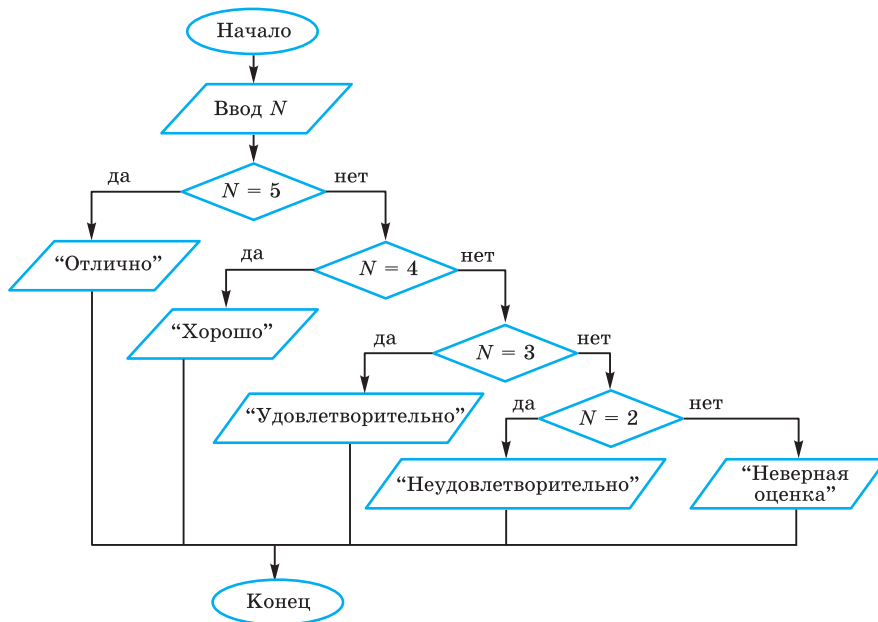


Рис. 3.14. Алгоритм перевода числовой оценки в словесную

Этот алгоритм имеет структуру вложенных ветвлений и может быть запрограммирован с использованием условного оператора **If** следующим образом:

```
Program Marks_1;
Var N: Integer;
Begin
    WriteLn('Введите оценку:');
    ReadLn(N);
    If N=5
    Then WriteLn('Отлично')
    Else If N=4
        Then WriteLn('Хорошо')
        Else If N=3
            Then WriteLn('Удовлетворительно')
            Else If N=2
                Then WriteLn('Неудовлетворительно')
                Else WriteLn('Неверная оценка')

End.
```

Пример 3. Решение рассмотренной в предыдущем примере задачи можно запрограммировать с помощью одного **оператора выбора**, имеющегося в языке Паскаль. Вот как будет выглядеть такая программа:

```
Program Marks_2;
Var N: Integer;
Begin
    WriteLn('Введите оценку:');
    ReadLn(N);
    Case N of
        5: WriteLn('Отлично');
        4: WriteLn('Хорошо');
        3: WriteLn('Удовлетворительно');
        2: WriteLn('Неудовлетворительно');
        Else WriteLn('Неверная оценка')
    End;
End.
```

Оператор выбора имеет следующий формат:

```

Case <селектор> of
    <список констант 1>: <оператор 1>;
    ...
    <список констант N>: <оператор N>;
    Else <оператор>
End
    
```

Здесь <селектор> — это выражение любого *порядкового типа*; <константа> — постоянная величина того же типа, что и селектор; <оператор> — любой простой или составной оператор.

Выполнение оператора выбора происходит так: вычисляется выражение-селектор; затем в списках констант ищется такое значение, которое совпадает с полученным значением селектора; далее исполняется оператор, помеченный данной константой. Если такой константы не найдено, то происходит переход к выполнению оператора, следующего после слова **Else**.

Пример 4. В этом примере демонстрируется использование списка констант в операторе выбора. Программа сообщает, сдал студент экзамен или не сдал. Если оценка одна из следующих: 3, 4, 5, то экзамен сдан; если 2, то не сдан.

```

Case N of
    3, 4, 5: WriteLn('Экзамен сдан');
    2: WriteLn('Экзамен не сдан');
    Else WriteLn('Нет такой оценки')
End
    
```

Так же как условный оператор, оператор выбора может использоваться в неполной форме, т. е. без ветви **Else**.

Если применить условный оператор, то эта программа запишется так:

```

If (N=3) or (N=4) or (N=5)
Then WriteLn('Экзамен сдан')
Else If N=2
    Then WriteLn('Экзамен не сдан')
    Else WriteLn('Нет такой оценки');
    
```

В условии ветвления использовано сложное логическое выражение, содержащее операции логического сложения **or** (или).



Система основных понятий

Программирование ветвлений	
Условный оператор	Оператор выбора
If <логическое выражение> Then <оператор 1> Else <оператор 2> Если <логическое выражение> истинно, то выполняется <оператор 1>, иначе выполняется <оператор 2>. <Оператор 1> и <оператор 2> — простые или составные операторы. В неполном ветвлении отсутствует ветвь Else	Case <селектор> of <список констант 1>: <оператор 1>; ... <список констант N>: <оператор N>; Else <оператор> End Здесь <селектор> — выражение порядкового типа; константы имеют тот же тип. Выполняется только одна из ветвей выбора, которая содержит константу, совпадающую со значением селектора. Ветвь Else может отсутствовать



Вопросы и задания

1. Какие операторы используются для программирования ветвящихся алгоритмов?
2. В каких случаях удобно использование оператора выбора?
3. Составьте на Паскале программу упорядочения по возрастанию значений в трех переменных: X, Y, Z.
4. Используя оператор выбора, составьте программу, которая по введенному номеру месяца будет выводить название соответствующего времени года (зима, весна, лето, осень).



§ 20

Пример поэтапной разработки программы решения задачи

Постановка задачи и формализация

Словом «задача» называют проблему, которая требует решения. Решение задачи начинается с ее постановки. На этапе *постановки задачи* в терминах предметной области (физики, экономики, биологии и др.) определяются исходные данные и результаты, которые надо получить.

Следующий этап — *формализация задачи*. Чаще всего процесс формализации означает перевод задачи на язык математики: формул, уравнений, неравенств, систем уравнений, систем неравенств и т. п.

Подробнее о формализации будет рассказано в разделе, посвященном информационному моделированию (в 11 классе). Некоторые представления об этом вы уже имеете из курса информатики для 7–9 классов.

Решение полученной математической задачи требует знания математики, умения выполнять *анализ математической задачи*. Такой анализ необходим для того, чтобы построить правильный алгоритм решения, обладающий всеми свойствами алгоритма.

Анализ математической задачи

Пусть в результате формализации некоторой задачи было получено квадратное уравнение: $ax^2 + bx + c = 0$, где коэффициенты a , b , c являются исходными данными. Требуется решить это уравнение, т. е. найти его корни. Проведем анализ этой математической задачи.

Рассмотрим различные варианты значений исходных данных, которые приводят к разным результатам для решающего ее алгоритма. Ограничимся только поиском вещественных корней уравнения. Проанализируем все возможные варианты множества значений коэффициентов a , b , c :

Если $a = 0$, $b = 0$, $c = 0$,	то любое x — решение уравнения
Если $a = 0$, $b = 0$, $c \neq 0$,	то уравнение решений не имеет
Если $a = 0$, $b \neq 0$,	то это линейное уравнение, которое имеет одно решение: $x = -c/b$
Если $a \neq 0$ и $d = b^2 - 4ac \geq 0$,	то уравнение имеет два вещественных корня: $x_1 = (-b + \sqrt{d})/(2a)$, $x_2 = (-b - \sqrt{d})/(2a)$
Если $a \neq 0$ и $d < 0$,	то уравнение не имеет вещественных корней



Построение алгоритма

Построим блок-схему алгоритма решения квадратного уравнения (рис. 3.15), учитывающего все ситуации, описанные в анализе задачи.

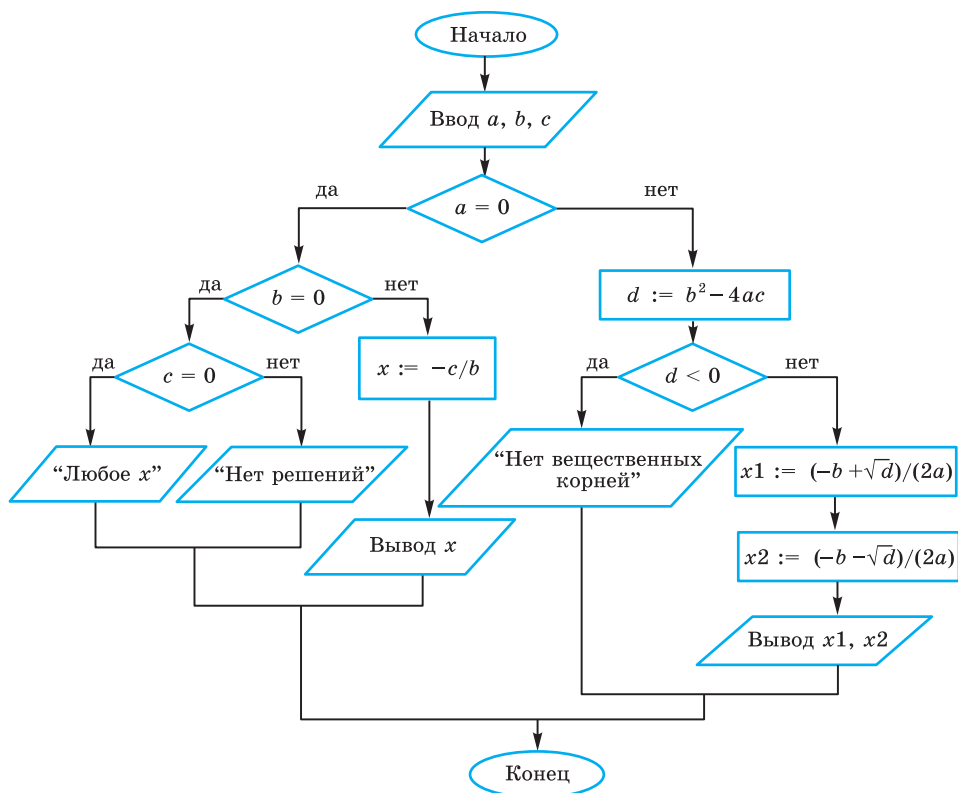


Рис. 3.15. Блок-схема алгоритма решения квадратного уравнения

Построенный алгоритм, несомненно, удовлетворяет свойству универсальности по отношению к исходным данным. Запишем этот же алгоритм на учебном Алгоритмическом языке.

```

алг корни квадратного уравнения
вещ a, b, c, d, x, x1, x2
нач ввод a, b, c
  если a=0
  то

```

```
если  $b=0$ 
то
  если  $c=0$ 
  то вывод “Любое  $x$  – решение”
  иначе вывод “Нет решений”
  все
иначе
   $x := -c/b$ 
  вывод  $x$ 
  все
иначе
   $d := b^2 - 4ac$ 
  если  $d < 0$ 
  то вывод “Нет вещественных корней”
  иначе
     $x1 := (-b + \sqrt{d})/(2a)$ ;  $x2 := (-b - \sqrt{d})/(2a)$ 
    вывод “ $x1=$ ”,  $x1$ , “ $x2=$ ”,  $x2$ 
  все
все
кон
```

Обратите внимание на смещения строк в тексте алгоритма — соблюдается принцип структуризации внешнего вида (§ 13). Повторим его: запись всякой вложенной структуры должна быть смещена на несколько позиций вправо относительно записи внешней структуры, а конструкции одного уровня вложенности записываются на одном вертикальном уровне.

Программирование

Алгоритмический язык (АЯ) — это язык описания алгоритмов с русскими служебными словами. После того как алгоритм записан на АЯ, составление программы на Паскале становится несложной задачей. Основное внимание следует уделять строгому соблюдению синтаксических правил языка. Правило смещения строк в тексте программы то же, что было сформулировано в § 13 для АЯ. Соответствующие друг другу служебные слова **Begin** и **End** должны располагаться друг под другом.

```
Program Roots;
Var a, b, c, d, x, x1, x2: Real;
```

```
Begin
WriteLn('Введите коэффициенты квадратного
        уравнения:');
Write('a='); ReadLn(a);
Write('b='); ReadLn(b);
Write('c='); ReadLn(c);
If a=0
Then
    If b=0
    Then
        If c=0
        Then WriteLn('Любое x - решение')
        Else WriteLn('Нет решений')
    Else
    Begin
        x:=-c/b;
        WriteLn('x=', x)
    End
Else
Begin
    d:=b*b-4*a*c;
    If d<0
    Then WriteLn('Нет вещественных корней')
    Else
    Begin
        x1:=(-b+sqrt(d))/2/a;
        x2:=(-b-sqrt(d))/2/a;
        WriteLn('x1=', x1);
        WriteLn('x2=', x2)
    End
End
End.
```

Чем больше текст программы, тем больше вероятность совершения ошибок при ее записи и вводе в компьютер. Ошибки, нарушающие правила грамматики языка, называются **синтаксическими ошибками**. Поиск и устранение синтаксических ошибок в программе называются **отладкой**. Отладить программу программисту помогает система программирования на данном языке, которая автоматически обнаруживает ошибки и сообщает о них программисту.

Тестирование программы

Тестирование — это этап, на котором экспериментально доказывается правильность алгоритма, заключенного в программе, и работоспособность программы. **Тест** — это вариант решения задачи с заданными исходными данными, для которых известен результат.

Предварительно должен быть составлен *план тестирования*. Для ветвящегося алгоритма должны быть протестированы все его ветви. В нашем примере пять ветвей, пять вариантов ответа. Значит, в плане тестирования должно быть не менее пяти вариантов теста.

В таблице 3.5 представлен план тестирования программы Roots и результаты проведенного тестирования.

Таблица 3.5

План и результаты тестирования

№	Исходные значения	Верные результаты	Результаты тестирования
1	$a = 0, b = 0, c = 0$	Любое X — решение	Любое X — решение
2	$a = 0, b = 0, c = 1$	Нет решений	Нет решений
3	$a = 0, b = 2, c = -6$	$x = 3$	$x=3$
4	$a = 2, b = 1, c = -3$	$x_1 = 1, x_2 = -1,5$	$x1=1 \quad x2=-1,5$
5	$a = -1, b = -1, c = -2$	Нет вещественных корней	Нет вещественных корней

Теперь, анализируя результаты тестирования, делаем вывод: правильность алгоритма и работоспособность программы доказаны.

Если какой-то из вариантов теста не дает ожидаемого результата, то в программе есть ошибки. Например, пусть программист ошибочно записал следующие операторы присваивания для вычисления корней:

$$x1 := (-b + \sqrt{d}) / 2 * a; \quad x2 := (-b - \sqrt{d}) / 2 * a;$$

Результаты всех тестов, кроме 4-го, совпали с ожидаемыми, а в 4-м тесте получилось: $x1=4, \quad x2=-6$. После этого программист обратит внимание на выражения для вычисления корней и исправит ошибки: либо заменит знак умножения на знак деления, либо заключит в скобки выражение $2 * a$.



Система основных понятий

Этапы решения задачи	
Постановка задачи	Определение исходных данных и искомых результатов (в терминах предметной области)
Формализация	Переход к задаче обработки некоторой знаковой системы, например к математической задаче
Анализ математической задачи	Определение всех вариантов множеств значений исходных данных. Определение для каждого варианта способа решения и вида выходных данных (результатов)
Построение алгоритма	Определение структуры алгоритма, последовательности команд. Представление на каком-либо языке описания алгоритмов (блок-схема, учебный Алгоритмический язык)
Составление программы	Запись и отладка программы на языке программирования. Строгое соблюдение правил синтаксиса языка
Тестирование	Экспериментальное доказательство правильности алгоритма и работоспособности программы. Тест — вариант решения задачи с заданными исходными данными, для которых известен результат. План тестирования строится так, чтобы наиболее полно проверить работу программы



Вопросы и задания



1. Сформулируйте основные цели этапов алгоритмического решения задачи.
2. Проанализируйте задачу решения биквадратного уравнения, составьте алгоритм и напишите программу на Паскале.



§ 21

Программирование циклов

Рассмотрим приемы программирования циклов на Паскале. В § 13 рассказывалось о том, что существуют две циклические алгоритмические структуры: цикл с предусловием (цикл-пока) и цикл с постусловием (цикл-до). Были показаны способы описания ци-

клических структур в блок-схемах и на Алгоритмическом языке. Форматы соответствующих операторов цикла в Паскале следующие.

Цикл с предусловием (цикл-пока):

While <логическое выражение> **Do**
 <оператор>

Цикл с постусловием (цикл-до):

Repeat
 <оператор>
Until <логическое выражение>

Различают циклы с заданным числом повторений и итерационные циклы.

На примерах конкретных задач рассмотрим приемы программирования циклов.

В математике известно, что сумма следующего бесконечного числового ряда:

$$1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{i!} + \dots$$

в пределе стремится к значению константы $e = 2,71828182\dots$. Функция e^x называется экспонентой, а логарифм по основанию e называется натуральным логарифмом: $\ln x$.

Требуется составить программу, вычисляющую эту константу по сумме числового ряда. Напомним, что символ «!» читается как «факториал» — функция, определенная следующим образом:

$$x! = \begin{cases} 1, & \text{при } x = 0, \\ 1 \cdot 2 \cdot 3 \cdot \dots \cdot x, & \text{при } x > 0. \end{cases}$$

Если слагаемые в вычисляемом выражении обозначить так:

$$a_0 = 1, a_1 = \frac{1}{1!} = \frac{1}{1}, a_2 = \frac{1}{2!} = \frac{1}{1 \cdot 2}, a_3 = \frac{1}{3!} = \frac{1}{1 \cdot 2 \cdot 3}, \dots,$$

то обобщенная формула для i -го элемента будет следующей:

$$a_i = \frac{1}{i!}.$$

Нетрудно увидеть, что между элементами данной последовательности имеется зависимость:

$$a_0 = 1, a_1 = \frac{a_0}{1}, a_2 = \frac{a_1}{2}, a_3 = \frac{a_2}{3} \text{ и т. д.}$$



Такая зависимость называется *рекуррентной зависимостью*, а соответствующая числовая последовательность — рекуррентной последовательностью. Данная рекуррентная последовательность может быть описана следующей ветвящейся формулой, которая называется рекуррентной формулой:

$$a_i = \begin{cases} 1, & \text{при } i = 0, \\ \frac{a_{i-1}}{i}, & \text{при } i > 0. \end{cases}$$

Циклы с заданным числом повторений

Пример 1. Дано целое положительное значение N . Требуется вычислить сумму:

$$1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{N!}.$$

Ниже приводятся два варианта программы решения этой задачи. В первом варианте используется цикл с предусловием, во втором — цикл с постусловием.

```

Program Summa_1;
Var E,a: Real; N,i: Integer;
Begin
  Write('N='); ReadLn(N);
  E:=0; i:=0; a:=1;
  While i<=N Do
    Begin
      E:=E+a;
      i:=i+1;
      a:=a/i
    End;
  WriteLn('E=', E)
End.

```

```

Program Summa_2;
Var E,a: Real; N,i: Integer;
Begin
  Write('N='); ReadLn(N);
  E:=0; i:=0; a:=1;
  Repeat
    E:=E+a;
    i:=i+1;
    a:=a/i;
  Until i>N;
  WriteLn('E=', E)
End.

```

Обратите внимание на то, как цикл с предусловием преобразуется в цикл с постусловием — условие цикла помещается после тела цикла и заменяется на противоположное:

$$\text{Not } (i \leq N) = i > N.$$

И тот, и другой цикл повторяют свое выполнение $(N + 1)$ раз. Переменная i выполняет роль не только знаменателя в дроби

$1/i!$, но и является счетчиком числа повторений цикла. Такие переменные называются **параметрами цикла**. И еще: в цикле с постусловием служебные слова **Repeat** и **Until** сами выполняют роль операторных скобок. Поэтому писать **Begin** и **End** здесь не требуется.

Выполнение этих программ на компьютере для значения $N = 7$ приводит к следующему результату: $E=2,7182539$.

Для программирования циклов с заданным числом повторений при постоянном шаге изменения параметра цикла в Паскале существует **цикл с параметром**. Вот как выглядит программа решения той же задачи с использованием цикла с параметром:

```
Program Summa_3;
Var E, a: Real; N, i: Integer;
Begin
  Write('N='); ReadLn(N);
  E:=1; a:=1;
  For i:=1 To N Do
    Begin
      a:=a/i;
      E:=E+a
    End;
  WriteLn('E=', E)
End.
```

В программе используется оператор цикла **For**, для которого существуют два варианта:

- 1) **For** <параметр цикла>:=<выражение 1> **To** <выражение 2>
 Do <оператор>
- 2) **For** <параметр цикла>:=<выражение 1> **Downto** <выражение 2>
 Do <оператор>

Здесь <параметр цикла> — имя простой переменной порядкового типа. Выполнение оператора **For** в первом варианте (**To**) происходит по следующей схеме.

1. Вычисляются значения <выражения 1> и <выражения 2>. Это делается только один раз при входе в цикл.
2. Параметру цикла присваивается значение <выражения 1>.
3. Значение параметра цикла сравнивается со значением <выражения 2>. Если параметр цикла меньше или равен этому значению, то выполняется тело цикла (<оператор>), в противном случае выполнение цикла заканчивается.

4. Значение параметра цикла изменяется на следующее значение в его типе (для целых чисел — увеличивается на единицу); происходит возврат к пункту 3.

Оператор цикла **For** объединяет в себе действия, которые при использовании цикла **While** выполняют различные операторы: присваивание параметру начального значения, сравнение его с конечным значением, изменение значения параметра на следующее.

Во втором варианте оператора **For** слово **Downto** буквально можно перевести как «вниз до». В таком случае параметр цикла изменяется по убыванию, т. е. при каждом повторении цикла параметр изменяет свое значение на предыдущее (равносильно `i:=pred(i)`).

Работая с оператором **For**, учитывайте следующие правила:

- параметр цикла не может иметь вещественного типа;
- в теле цикла нельзя изменять переменную-параметр цикла;
- при выходе из цикла значение переменной-параметра является неопределенным.

Одной из особенностей языка PascalABC.NET является свойственное многим современным языкам программирования стремление описывать параметр цикла непосредственно в заголовке цикла, тем самым локализуя параметр в этом цикле. В существующих версиях PascalABC.NET это свойство цикла **For** выражено непоследовательно. Иногда компилятор позволяет писать программы так, как это было сделано в примере 1, а иногда (как правило, когда переменная цикла используется не только в одном цикле) возникает сообщение «Параметр цикла for в PascalABC.NET должен описываться в заголовке цикла» и компиляция не проходит. В последнем случае надо заменить в приведенном выше примере фрагмент

```
For i:=1 To N Do
```

на фрагмент

```
For var i:=1 To N Do
```

и удалить переменную `i` из раздела описаний.

Рассмотрим пример программы, в которой в теле цикла будет присутствовать ветвление.

Пример 2. Составим программу проверки знаний учеником таблицы умножения. Компьютер задает ученику 10 вопросов на умножение чисел от 2 до 9. На каждое задание ученик вводит свой ответ, а компьютер сообщает, верный ответ или нет.

На рисунке 3.16 приведена блок-схема такого алгоритма.

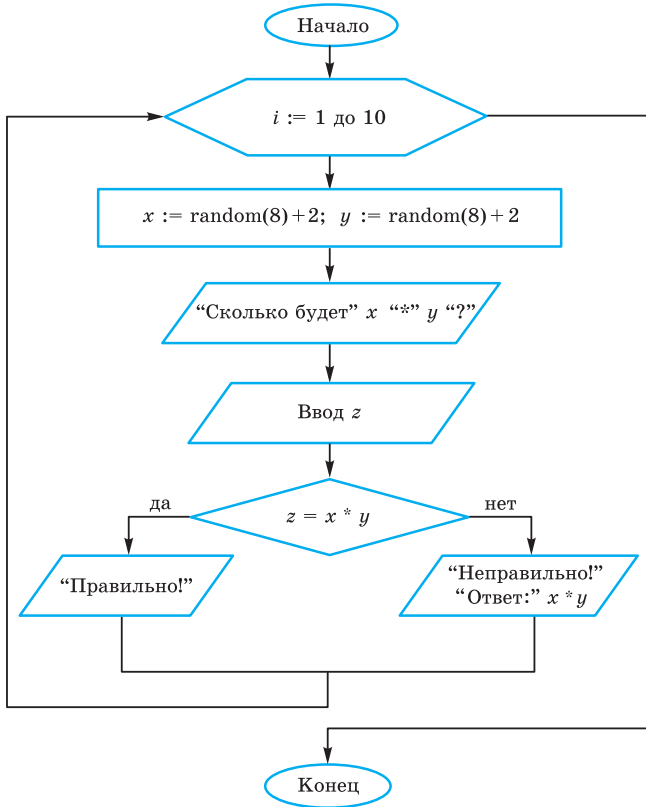


Рис. 3.16. Блок-схема алгоритма из примера 2

Обратите внимание на то, как отображается на блок-схеме цикл с параметром.

В этом алгоритме использована функция `random(x)`, результатом выполнения которой является случайное целое число из диапазона от 0 до $x - 1$. Следовательно, выражение `random(8)+2` принимает случайные значения от 2 до 9. Функция `random` называется датчиком случайных чисел.

На Паскале этот алгоритм программируется так:

```

Program TablMul;
Begin
  Var x,y,i,z: Integer;
  randomize;
  For i:=1 To 10 Do
    Begin

```

```

x:=random(8)+2;
y:=random(8)+2;
WriteLn('Сколько будет ', x, '*', y, '?');
Read(z);
If z=x*y
Then WriteLn('Правильно!')
Else WriteLn('Неправильно! ', x, '*', y, '=', x*y);
End;
End.

```

А вот фрагмент интерфейса исполнения этой программы:

```

Сколько будет 4*8?
21
Неправильно! 4*8=32
Сколько будет 6*9?
54
Правильно!

```

В программе используется стандартная процедура `randomize`. Ее исполнение производит установку случайного начального состояния датчика случайных чисел. Благодаря этому при повторном выполнении программы будут получаться разные последовательности случайных чисел.

Система основных понятий

Программирование циклов		
Операторы цикла		
Цикл-пока	Цикл с параметром	Цикл-до
While <логическое выражение> Do <оператор> <оператор> — тело цикла. Цикл повторяет выполнение, пока истинно <логическое выражение>	Параметр — переменная порядкового типа. 1) For <параметр цикла>:= <выражение 1> To <выражение 2> Do <оператор> — по возрастанию параметра 2) For <параметр цикла>:= <выражение 1> Downto <выражение 2> Do <оператор> — по убыванию параметра	Repeat <оператор> Until <логическое выражение> <оператор> — тело цикла. Повторяется выполнение тела цикла до того, как <логическое выражение> станет истинным
Цикл с заданным числом повторений имеет управляющий параметр, изменяющийся с постоянным шагом в определенном диапазоне значений; реализуется всеми типами операторов цикла		

Вопросы и задания

1. Постройте трассировочную таблицу выполнения программы Summa_1 для значения $n = 3$.
2. Составьте программу, по которой компьютер десять раз запросит ввод любых чисел и в результате выведет среднее арифметическое значение введенных чисел (массив не использовать). Сделайте три варианта программы, используя операторы цикла **While**, **Repeat** и **For**.
3. Составьте программу, по которой компьютер десять раз запросит ввод целых чисел и в результате сообщит, сколько среди введенных чисел четных и сколько нечетных.
4. Составьте программу, по которой на экран будет выведена вся таблица умножения.
5. Усовершенствуйте программу TablMul (пример 2 данного параграфа) таким образом, чтобы в результате выводилось сообщение о количестве правильных и неправильных ответов.
6. Усовершенствуйте программу TablMul таким образом, чтобы в результате выводилась оценка, поставленная ученику. Критерии для выставления оценок придумайте сами.
7. Получите таблицу значений функции $\sin x$ и $\cos x$ на отрезке $[0, 1]$ с шагом 0,1 в следующем виде:

x	$\sin x$	$\cos x$
0.0000	0.0000	1.0000
0.1000	0.0998	0.9950
...	...	...
1.0000	0.8415	0.5403

8. Получите в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр.
9. Значение функции e^x (экспонента от x) равно сходящейся сумме бесконечного ряда:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!} + \dots$$

Получите рекуррентную формулу для слагаемых. Используя операторы цикла **While**, **Repeat** и **For**, составьте три варианта программы вычисления суммы с заданным числом слагаемых.

§ 22

Вложенные и итерационные циклы

Если в теле одного цикла имеется другой цикл, то такая структура алгоритма называется **вложенными циклами**. Рассмотрим задачу, программа решения которой имеет структуру вложенных циклов.

Требуется получить на экране компьютера таблицу умножения в форме матрицы Пифагора. Блок-схема алгоритма будет следующей (рис. 3.17).

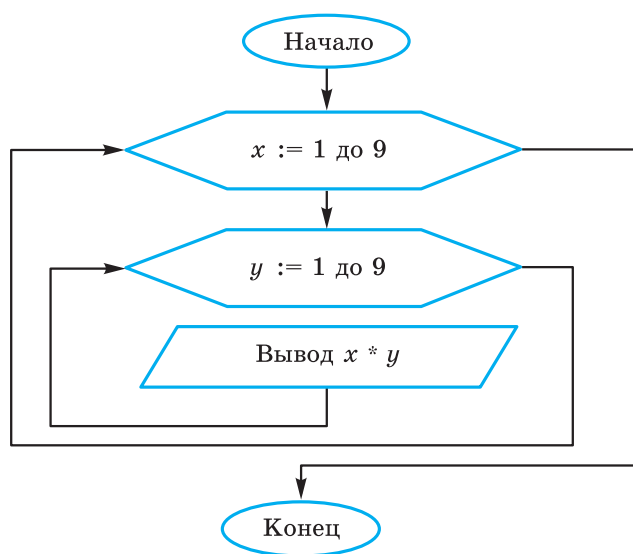


Рис. 3.17. Блок-схема алгоритма получения таблицы умножения

Здесь цикл по параметру y вложен в цикл по параметру x . Последовательность изменения значений параметров циклов такая:

$$\begin{aligned} x = 1; & \quad y = 1, 2, 3, \dots, 9 \\ x = 2; & \quad y = 1, 2, 3, \dots, 9 \\ & \dots \\ x = 9; & \quad y = 1, 2, 3, \dots, 9 \end{aligned}$$

Таким образом, внешний цикл исполнится 9 раз, а внутренний — $9 \cdot 9 = 81$ раз. На один шаг повторения внешнего цикла происходит полная прокрутка внутреннего.

При программировании вложенных циклов используется понятие глубины вложенности. В данном примере глубина вложенности внутреннего цикла равна единице. Если бы внутри вложенного цикла был еще один вложенный цикл (например, для вычисления всех вариантов перемножения трех сомножителей), то его глубина вложенности равнялась бы двум.

Программа на Паскале получения матрицы Пифагора:

```
Program MatrPif;  
Var x, y: 1..9;  
Begin  
  For x:=1 To 9 Do  
    Begin  
      WriteLn;  
      For y:=1 To 9 Do  
        Write(x*y:3)  
      End  
    End  
  End.
```

В результате выполнения программы на экране получим:

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

В программе присутствуют некоторые элементы, не отраженные в блок-схеме. В описании переменных X , Y использован ограниченный тип: $1..9$, поскольку в данной задаче величины принимают целые значения только в этом диапазоне. Оператор `WriteLn` перед началом вложенного цикла обеспечивает переход к новой строке в таблице каждый раз при смене первого множителя. В операторе `Write(X*Y:3)` для вывода значения произведения после двоеточия использован указатель формата — 3 . Это обеспечивает вывод чисел в три позиции на экране, благодаря чему соответствующие столбцы таблицы располагаются строго друг под другом. Первая строка и первый столбец на экране — это множители, что соответствует стандартному формату таблицы Пифагора.

Итерационные циклы

Итерационный цикл — это цикл, для которого число повторений тела цикла заранее неизвестно.

В итерационных циклах на каждом шаге вычислений происходят последовательное приближение и проверка условия достижения искомого результата.

Выход из итерационного цикла осуществляется в случае выполнения заданного условия.

Пример 1. Снова рассмотрим задачу вычисления суммы числового ряда:

$$1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{i!} + \dots$$

Но теперь условие будет таким: в сумму нужно включить только слагаемые, значение которых больше некоторой малой величины ε . При этом полученная сумма будет отличаться от предельного значения (константы e) на величину, не большую ε .

Поскольку с увеличением значения i величина $1/i!$ уменьшается, в сумму надо включать все слагаемые, предшествующие первому значению, меньшему ε .

Вот две программы решения этой задачи, использующие циклы с предусловием и постусловием:

```

Program Summa_4;
Var E,a,eps: Real;
      i:Integer;
Begin
  Write('Eps=');
  ReadLn(eps);
  E:=0; i:=0; a:=1;
  While a>eps Do
    Begin
      E:=E+a;
      i:=i+1;
      a:=a/i
    End;
  WriteLn('E=', E,
          'Слагаемых:', i)
End.

```

```

Program Summa_5;
Var E,a,eps: Real;
      i:Integer;
Begin
  Write('Eps=');
  ReadLn(eps);
  E:=0; i:=0; a:=1;
  Repeat
    E:=E+a;
    i:=i+1;
    a:=a/i;
  Until a<=eps;
  WriteLn('E=', E,
          'Слагаемых:', i)
End.

```

Решить эту задачу, используя цикл с параметром, нельзя. Итерационные циклы программируются путем использования либо цикла-пока, либо цикла-до.

В качестве результата выводится значение суммы и число вошедших в нее слагаемых. Выполнение этих программ для значения $\varepsilon = 10^{-8}$ дает в результате: E=2,71828182 Слагаемых: 12. Таким образом, за 12 повторений цикла значение константы e получено с точностью до 8 знаков после запятой. Слово «итерации» означает «приближения». С каждым повторением цикла вычисляемая величина приближалась к предельному значению константы.

Пример 2. В § 17 была рассмотрена задача вычисления суммы цифр трехзначного натурального числа. Программа имела линейную структуру. Поставим задачу в более общем виде: для любого многозначного натурального числа вычислить сумму всех его цифр.

Выделение цифр происходит с помощью однотипных действий: использования операций `mod` и `div`. Очевидно, что их можно «зациклить». Однако число повторений цикла будет разным для чисел разной длины. Поэтому эта задача не решается с помощью цикла с заданным числом повторений. В таком случае в программе можно использовать либо оператор цикла **While**, либо **Repeat** и нельзя — цикл с параметром **For**.

Программа с использованием цикла с предусловием:

```

Program SumCifr;
var X: Longint; Sum: Word;
Begin
    Write('Введите целое число: '); ReadLn(X);
    Sum:=0;
    While (X>0) Do
        Begin
            Sum:=Sum + X mod 10; //прибавление к сумме
                                //младшей цифры
            X:=X div 10          //отбрасывание в X
                                //младшей цифры
        End;
    WriteLn('Сумма цифр = ', Sum)
End.

```

Поскольку при каждом повторении цикла от числа X отбрасывается одна младшая цифра, закончить цикл нужно тогда, когда X станет равным нулю. Обратите внимание на типы переменных. Надо помнить о разнообразии групп типов в Паскале. Назначение переменной X типа `Longint` дает возможность вводить в нее значения, включающие до десяти знаков. Для переменной `Sum` назначен тип `Word`, поскольку сумма цифр может быть только положительным числом.



Система основных понятий

Вложенные и итерационные циклы

Вложенный цикл — цикл, входящий в тело другого (внешнего) цикла.

На каждом шаге внешнего цикла происходит полная прокрутка внутреннего цикла.

Чем больше глубина вложенности цикла, тем большее число раз он выполняется.

Итерационный цикл — это цикл, для которого число повторений тела цикла заранее неизвестно. Итерационные циклы программируются с помощью операторов **While** и **Repeat**

Вопросы и задания

1. Найдите все значения натуральных чисел X, Y, Z из интервала от 1 до 10, удовлетворяющих равенству: $X^2 + Y^2 = Z^2$.
2. Вычислите количество точек с целочисленными координатами, попадающих в круг радиуса R ($R > 0$) с центром в начале координат.
3. *Старинная задача.* Сколько можно купить быков, коров и телят на 100 руб., если в сумме должно быть куплено 100 голов скота, а цена быка — 10 руб., цена коровы — 5 руб., цена теленка — 0,5 руб.?
4. Чем отличается итерационный цикл от цикла с заданным числом повторений?
5. Почему для программирования итерационных циклов не используется оператор цикла с параметром?
6. Запрограммируйте итерационный цикл вычисления функции e^x (см. задание 9 из § 21) с точностью ε . Сделайте два варианта программы: с циклами **While** и **Repeat**. Выполните вычисления для $\varepsilon = 10^{-6}$, $x = 2$ и сопоставьте полученный результат со значением стандартной функции $\exp(x)$.
7. Составьте программу определения количества четных и нечетных цифр в записи данного натурального числа.
8. Составьте программу определения наибольшей цифры в записи данного натурального числа.

§ 23**Вспомогательные алгоритмы
и подпрограммы**

Еще одним важнейшим методологическим приемом структурного программирования является *декомпозиция решаемой задачи на подзадачи* — более простые, с точки зрения программирования, части исходной задачи. Алгоритмы решения таких подзадач называются **вспомогательными алгоритмами**.

В языках программирования вспомогательные алгоритмы называются **подпрограммами**. В Паскале различаются две разновидности подпрограмм: **процедуры** и **функции**. Рассмотрим этот вопрос на примере следующей задачи: даны два натуральных числа a и b . Требуется определить наибольший общий делитель трех величин: $a + b$, $a^2 + b^2$, $a \cdot b$. Запишем это так: $\text{НОД}(a + b, a^2 + b^2, a \cdot b)$.

Идея решения состоит в следующем математическом факте: если x, y, z — три натуральных числа, то $\text{НОД}(x, y, z) = \text{НОД}(\text{НОД}(x, y), z)$. Иначе говоря, нужно найти НОД двух величин, а затем НОД полученного значения и третьего числа (попробуйте это доказать).

Очевидно, что вспомогательным алгоритмом для решения поставленной задачи является алгоритм получения наибольшего общего делителя двух чисел. Эта задача решается с помощью алгоритма Евклида, который подробно обсуждался в 9 классе. Напомним, что идея алгоритма Евклида основана на следующей формуле:

$$\text{НОД}(M, N) = \begin{cases} M, & \text{при } M = N; \\ \text{НОД}(M - N, N), & \text{при } M > N; \\ \text{НОД}(N, N - M), & \text{при } N > M. \end{cases}$$

Приведем алгоритм решения поставленной задачи на учебном Алгоритмическом языке. Алгоритм состоит из процедуры «Евклид» и основного алгоритма «Задача», в котором присутствуют два обращения к процедуре:

процедура Евклид(цел M, N, K)

нач

пока $M <> N$

нц

если $M > N$

то $M := M - N$

иначе $N := N - M$

все

кц

$K := M$

кон

алг задача

цел a, b, c

нач

ввод(a, b)

Евклид($a+b, a*a+b*b, c$)

Евклид($c, a*b, c$)

вывод(c)

кон

Здесь M , N и K являются формальными параметрами процедуры. M и N — параметры-аргументы, K — параметр-результат.

Процедуры в Паскале. Основное отличие процедур в Паскале от процедур в Алгоритмическом языке (АЯ) состоит в том, что процедуры в Паскале описываются в разделе описания подпрограмм, а в АЯ процедура является внешней по отношению к вызывающей программе. Теперь посмотрим, как решение поставленной задачи программируется на Паскале.

```
Program NOD1;
Var A, B, C: Integer;
Procedure Evklid(M, N: Integer; Var K: Integer);
Begin
    While M<>N Do
        If M>N
            Then M:=M-N
            Else N:=N-M;
    K:=M
End;
Begin
    Write('a = '); ReadLn(A);
    Write('b = '); ReadLn(B);
    Evklid(A+B, A*A+B*B, C);
    Evklid(C, A*B, C);
    WriteLn('НОД = ', C)
End.
```

В данном примере **обмен** аргументами и результатами между основной программой и процедурой производится **через параметры**. Описание процедуры на Паскале имеет следующий формат:

```
Procedure <имя процедуры> [(список формальных параметров)];
    <блок>
```

Квадратные скобки указывают на то, что список формальных параметров может отсутствовать, т. е. возможна процедура без параметров. Параметры могут быть параметрами-переменными и параметрами-значениями. Параметры-переменные записываются следующим образом:

```
Var <список переменных>: <тип>
```

Параметры-значения указываются так:

```
<список переменных>: <тип>
```

Чаще всего аргументы представляются как параметры-значения (хотя могут быть и параметрами-переменными). А для передачи результатов используются параметры-переменные. Процедура в качестве результата может передавать в вызывающую программу множество значений (в частном случае — одно), а может и ни одного. Теперь рассмотрим правила обращения к процедуре. Обращение к процедуре производится в форме **оператора процедуры**:

<имя процедуры> [(список фактических параметров)]

Если описана процедура с формальными параметрами, то обращение к ней производится оператором процедуры с фактическими параметрами. Правила соответствия между формальными и фактическими параметрами: соответствие *по количеству*, соответствие *по последовательности* и соответствие *по типам*.

Взаимодействие формальных и фактических параметров через параметры-переменные называется передачей **по ссылке**: при обращении к процедуре ей передается ссылка на переменную, заданную в качестве фактического параметра. Эта ссылка и используется процедурой для доступа к этой переменной.

Другой вариант взаимодействия формальных и фактических параметров называется передачей **по значению**: вычисляется значение фактического параметра (выражения), и это значение присваивается соответствующему формальному параметру.

В рассмотренном нами примере формальные параметры M и N являются параметрами-значениями. Это аргументы процедуры. При обращении к ней первый раз им соответствуют значения выражений $A + B$ и $A * A + B * B$, второй раз — C и $A * B$. Параметр K является параметром-переменной. В ней получается результат работы процедуры. В обоих обращениях к процедуре соответствующим фактическим параметром является переменная C . Через эту переменную основная программа получает результат.

Теперь рассмотрим другой вариант программы, решающей ту же задачу. В ней используется процедура **без параметров**.

```
Program NOD2;
Var A, B, K, M, N: Integer;
Procedure Evklid;
Begin
  While M<>N Do
    If M>N
      Then M:=M-N
```

```
      Else N:=N-M;  
      K:=M  
End;  
Begin  
  Write('a= '); ReadLn(A);  
  Write('b= '); ReadLn(B);  
  M:=A+B; N:=A*A+B*B;  
  Evklid;  
  M:=K; N:=A*B;  
  Evklid;  
  WriteLn('НОД = ', K)  
End.
```

Чтобы разобраться в этом примере, требуется объяснить новое для нас понятие: область действия описания.

Областью действия описания любого программного объекта (переменной, типа, константы и т. д.) является тот блок, на который это описание распространяется. Если данный блок вложен в другой (подпрограмма), то присутствующие во вложенном блоке описания являются **локальными**. Они действуют только в пределах внутреннего блока. Описания же, расположенные во внешнем блоке, называются **глобальными** по отношению к внутреннему блоку. Если глобально описанный объект используется во внутреннем блоке, то на него распространяется внешнее (глобальное) описание.

В программе NOD1 переменные M , N , K являются локальными внутри процедуры; переменные A , B , C — глобальные. Однако внутри процедуры переменные A , B , C не используются. Связь между внешним блоком и процедурой осуществляется через параметры.

В программе NOD2 все переменные являются глобальными. В процедуре Evklid нет ни одной локальной переменной (нет и параметров). Переменные M и N , используемые в процедуре, получают свои значения через оператор присваивания в основном блоке программы и изменяют значения в подпрограмме. Результат получается в глобальной переменной K , значение которой выводится на экран. Здесь **обмен** значениями между основной программой и процедурой производится **через глобальные переменные**.

Использование механизма передачи через параметры делает процедуру более универсальной, независимой от основной программы. Однако в некоторых случаях оказывается удобнее

использовать передачу через глобальные переменные. Чаще такое бывает с процедурами, работающими с большими объемами информации. В этой ситуации глобальное взаимодействие экономит память компьютера.

Функции. Теперь выясним, что такое подпрограмма-функция. Обычно функция используется в том случае, когда результатом работы подпрограммы должна быть скалярная (простая) величина. Тип результата называется типом функции. Формат описания функции следующий:

```
Function <имя функции> [ (<список формальных
                          параметров> ) ] : <тип функции>;
      <блок>
```

Как и у процедуры, у функции в списке формальных параметров могут присутствовать параметры-переменные и параметры-значения. Всё это — аргументы функции. Параметры вообще могут отсутствовать, если аргументы передаются глобально.

Программа решения рассмотренной выше задачи с использованием функции будет выглядеть следующим образом:

```
Program NOD3;
Var A, B, Rez: Integer;
Function Evklid(M, N: Integer): Integer;
Begin
    While M<>N Do
        If M>N
            Then M:=M-N
            Else N:=N-M;
    Evklid:=M
End;
Begin
    Write('a = '); ReadLn(A);
    Write('b = '); ReadLn(B);
    Rez:=Evklid(Evklid(A+B, A*A+B*B), A*B);
    WriteLn('НОД = ', Rez)
End.
```

Из примера видно, что тело функции отличается от тела процедуры только тем, что в функции *результат присваивается идентификатору функции*: Evklid:=M.

Обращение к функции является операндом в выражении. Оно записывается в следующей форме:

<имя функции> (<список фактических параметров>)

Правила соответствия между формальными и фактическими параметрами все те же. Сравнивая приведенные выше программы, можно сделать вывод, что программа NOD3 имеет определенные преимущества перед другими. Функция позволяет получить результат путем выполнения одного оператора присваивания. Здесь также иллюстрируется возможность того, что фактическим аргументом при обращении к функции может быть эта же функция.

По правилам стандарта Паскаля, возврат в вызывающую программу из подпрограммы происходит, когда выполнение подпрограммы доходит до ее конца (последний **End**). Однако в современных версиях Паскаля есть средство, позволяющее выйти из подпрограммы в любом ее месте. Это оператор-процедура **Exit**. Например, функцию определения большего из двух данных вещественных чисел можно описать так:

```
Function Max(X, Y: Real): Real;  
Begin  
  Max:=X;  
  If X>Y Then Exit Else Max:=Y  
End;
```

Модифицированный алгоритм Евклида. Подпрограмму алгоритма Евклида можно составить иначе, если воспользоваться операцией **mod** (получение остатка от деления), имеющейся в Паскале. Идея алгоритма исходит из справедливости следующих равенств:

$$\text{НОД}(M, N) = \begin{cases} M, & \text{при } N=0; \\ \text{НОД}(N, M \bmod N), & \text{при } N \neq 0. \end{cases}$$

В таком случае функцию **Evklid** можно переписать так:

```
Function Evklid(M, N: Integer): Integer;  
Var R: Integer;  
Begin  
  While N<>0 Do  
    Begin R:=M mod N; M:=N; N:=R End;  
  Evklid:=M  
End;
```



Система основных понятий

Подпрограммы	
Процедуры	Функции
Результат — <i>любое число величин</i> Описание: Procedure <имя процедуры> [(список формальных параметров)] ; <блок>	Результат — <i>одна величина</i> Описание: Function <имя функции> [(<список формальных параметров>)] : <тип функции>; <блок>
Обращение — <i>оператор процедуры</i>: <имя процедуры> [(список фактических параметров)]	Обращение — <i>операнд выражения</i>: <имя функции> (<список фактических параметров>)
Параметры подпрограмм	
Параметры-переменные	Параметры-значения
Описание: Var <список переменных> : <тип>	Описание: <список переменных> : <тип>
Фактические параметры: <i>переменные</i>	Фактические параметры: <i>выражения</i>



Вопросы и задания

1. Для чего используются подпрограммы?
2. В чем различие между процедурами и функциями?
3. Какие существуют способы передачи данных между подпрограммой и вызывающей ее программой?
4. Составьте программу вычисления площади кольца по значениям внутреннего и внешнего радиусов, используя подпрограмму вычисления площади круга (два варианта: с процедурой и с функцией).
5. Составьте программу сложения двух простых дробей. Результат должен быть несократимой дробью. Используйте подпрограмму вычисления НОД по алгоритму Евклида. Простая дробь задается двумя целыми числами: числителем и знаменателем.



6. По координатам вершин треугольника вычислите его периметр, используя подпрограмму вычисления длины отрезка между двумя точками.
7. Даны три целых числа. Определите, у которого из них больше сумма цифр. Подсчет суммы цифр организуйте через подпрограмму.



§ 24

Массивы

Массивом в Паскале называют переменную величину регулярного типа.

Регулярный тип — это структурный тип данных, представляющих собой совокупность пронумерованных однотипных величин.

Описание массивов. Переменная регулярного типа описывается в разделе описания переменных в следующей форме:

```
Var <идентификатор>: array [<тип индекса>]  
                        of <тип компонентов>
```

В данном случае квадратные скобки — это обязательные символы, которые называются индексными скобками. Чаще всего в качестве типа индекса употребляется ограниченный тип. Например, массив вещественных чисел, хранящий 12 значений средне-месячных температур в течение года, опишется так:

```
Var T: array[1..12] of Real;
```

Описание массива определяет, во-первых, размещение массива в памяти, во-вторых, правила его дальнейшего употребления в программе.

Элемент массива идентифицируется в виде переменной с индексами:

```
<идентификатор массива> [<индексы элемента>]
```

Для одномерного массива индекс — это одно значение. Для многомерных массивов индекс — множество значений. В качестве индекса может употребляться любое выражение соответствующего типа. Например, для элементов массива температур возможны обозначения: $T[5]$, $T[k]$, $T[i+j]$, $T[m \text{ div } 2]$.

Последовательные элементы массива располагаются в последовательных ячейках памяти ($T[1]$, $T[2]$ и т. д.), причем значения индекса не должны выходить за диапазон $1..12$.

Тип индекса может быть любым скалярным порядковым типом, кроме `Integer`. Например, в программе могут присутствовать следующие описания:

```
Var cod: array[Char] of 1..100;  
      L: array[Boolean] of Char;
```

В такой программе допустимы следующие обозначения элементов массивов:

```
cod['x'];  L[true];  cod[chr(65)];  L[a>0].
```

В некоторых случаях бывает удобно в качестве индекса использовать перечислимый тип. Например, данные о количестве учеников в четырех десятых классах одной школы могут храниться в следующем массиве:

```
Type Index = (A, B, C, D);  
Var class_10: array[Index] of Byte;
```

И если, например, элемент `class_10[A]` равен 35, то это означает, что в 10А классе 35 человек. Такое индексирование улучшает наглядность программы.

Часто структурному типу присваивается имя в разделе типов, которое затем используется в разделе описания переменных.

```
Type Mas1 = array [1..100] of Integer;  
      Mas2 = array [-10..10] of Char;  
Var Num: Mas1; Sim: Mas2;
```

До сих пор речь шла об одномерных массивах, в которых типы элементов скалярные.

Многомерный массив в Паскале трактуется как одномерный массив, тип элементов которого также является массивом (массив массивов).

В качестве примера рассмотрим таблицу с информацией о среднемесячных температурах за 10 лет, например с 2001 по 2010 год. Очевидно, для этого удобна прямоугольная (двумерная) таблица, в которой столбцы соответствуют месяцам, а строки — годам.

Год	Месяц											
	1	2	3	4	5	6	7	8	9	10	11	12
2001	-23	-17	-8,4	6,5	14	18,6	25	19	12,3	5,6	-4,5	-19
2002	-16	-8,5	-3,2	7,1	8,4	13,8	28,5	21	6,5	2	-13	-20
2003	-9,8	-14	-9,2	4,6	15,6	21	17,8	20	11,2	8,1	-16	-21
...	...	...	...	...	...	...	...	...	...	...	...	...
2010	-25	-9,7	-3,8	8,5	13,9	17,8	23,5	17,5	10	5,7	-14	-20

Для обработки такой таблицы в программе следует описать массив:

```
Var Tabl: array[2001..2010] of array[1..12] of Real;
```

Вот примеры обозначения некоторых элементов этого массива:

```
Tabl[2001][1]; Tabl[2005][10]; Tabl[2010][12].
```

Однако чаще употребляется другая, эквивалентная форма обозначения элементов двумерного массива:

```
Tabl[2001,1]; Tabl[2005,10]; Tabl[2010,12].
```

Переменная Tabl[2001] обозначает всю первую строку таблицы, т. е. весь массив температур за 2001 год. Другим эквивалентным вариантом приведенному выше описанию является следующее:

```
Type Month = array [1..12] of Real;
```

```
Year = array [2001..2010] of Month;
```

```
Var Tabl: Year;
```

Наиболее краткий вариант описания данного массива такой:

```
Var Tabl: array[2001..2010, 1..12] of Real;
```

Продолжая по аналогии, можно определить трехмерный массив как одномерный массив, у которого элементами являются двумерные массивы. Вот пример описания трехмерного массива:

```
Var A: array[1..10, 1..20, 1..30] of Integer;
```

Это массив, состоящий из $10 \cdot 20 \cdot 30 = 6000$ целых чисел и занимающий в памяти $6000 \cdot 2 = 12\,000$ байтов. В Паскале нет ограничения сверху на размерность массива. Однако в каждой конкретной реализации Паскаля ограничивается объем памяти, выделяемый под массивы.

Чтобы упростить изменения размеров массива и всех операторов программы, связанных с этими размерами, удобно определять индексные параметры в разделе констант:

```
Const Imax = 10; Jmax = 20;  
Var Mas: array[1..Imax, 1..Jmax] of Integer;
```

Теперь для изменения размеров массива Mas и всех операторов программы, связанных с этими размерами, достаточно отредактировать только одну строку в программе — раздел констант.

В современных версиях Паскаля, включая PascalABC.NET, появились и динамические массивы, т. е. такие, размер которых определяется в процессе выполнения программы.

Отметим, что по аналогии с математикой одномерные числовые массивы часто называют векторами, а двумерные — матрицами.

Действия над массивом как единым целым. Такие действия допустимы лишь в двух случаях:

- присваивание значений одного массива другому;
- применение к массивам операций отношения «равно», «не равно».

В обоих случаях массивы должны иметь одинаковые типы (тип индексов и тип элементов).

Пример 1

```
Var P, Q: array[1..5, 1..10] of Real;
```

При выполнении операции присваивания

```
P:=Q
```

все элементы массива *P* станут равными соответствующим элементам массива *Q*.

Пример 2

Как уже отмечалось, в многомерных массивах переменная с индексом может обозначать целый массив. Тогда если массив `Tabl` описан так:

```
Type mas = array [1..12] of Real;  
Var Tabl: array[2001..2010] of mas;
```

и в нем требуется данные за 2009 год сделать такими же, как за 2001 год (девятой строке присвоить значение первой строки), то это можно сделать одним присваиванием:

```
Tabl[2009]:=Tabl[2001]
```

А если нужно поменять местами значения этих строк, то это делается через третью переменную того же типа:

```
P:=Tabl[2009]; Tabl[2009]:=Tabl[2001]; Tabl[2001]:=P;
```

где `P` описана так:

```
Var P: mas;
```

Ввод и вывод массивов производятся покомпонентно. Вот примеры ввода с клавиатуры значений одномерного и двумерного массивов:

```
For I:=1 To 12 Do  
  ReadLn(T[I]);  
  
For I:=1 To Imax Do  
  For J:=1 To Jmax Do  
    ReadLn(Mas[I, J]);
```

Здесь каждое следующее значение будет вводиться с новой строки. Для построчного ввода используется оператор `Read`.

Аналогично в цикле по индексной переменной организуется вывод значений массива на экран. Например:

```
For I:=1 To 12 Do Write(T[I]:8:4);
```

Напомним, что модификатор формата `8:4` означает вывод числа в формате с фиксированной точкой в 8 позициях, из которых в 4 последних позициях размещается дробная часть.

Следующий фрагмент программы организует построчный вывод матрицы на экран:

```

For I:=1 To Imax Do
Begin
    For J:=1 To Jmax Do
        Write (Mas[I,J]:6);
    WriteLn
End;

```

После вывода очередной строки матрицы оператор WriteLn без параметров переведет курсор в начало новой строки. Следует заметить, что в последнем примере матрица на экране будет получена в естественной форме прямоугольной таблицы, если Jmax не превышает 12 (подумайте почему).



Система основных понятий

Массивы	
Массив — переменная величина регулярного типа	
Регулярный тип — структурный тип данных, представляющих собой совокупность пронумерованных однотипных величин	
Описание массива	Идентификация элементов массива
Var <идентификатор>: array [<тип индекса>] of <тип компонентов> <тип индекса> — любой поряд- ковый тип, кроме Integer; тип компонентов — любой простой или структурный тип	<идентификатор массива> [<индексы элемента>] Для одномерного массива индекс — одно значение, для многомерного массива — множество значений
Действия над массивом как единым целым	
Присваивание однотипных массивов	Отношения «равно», «не равно» для однотипных массивов
Ввод/вывод массивов производится покомпонентно	

Вопросы и задания

1. Что такое регулярный тип данных? Что такое массив?
2. Какие типы допустимы для индексов массива?
3. Как в Паскале трактуется многомерный массив?
4. Какие действия можно выполнять над массивом как единым целым?
5. Дан вектор $\{z_i\}$, $i = 1, \dots, 50$. Составьте программу ввода значений и вычисления длины этого вектора по следующей формуле:

$$L = \sqrt{z_1^2 + z_2^2 + \dots + z_{50}^2}.$$

6. Даны значения массива $\{a_i\}$, $i = 0, \dots, 10$ и переменной x . Составьте программу вычисления алгебраического многочлена 10-й степени по формуле Горнера:

$$a_{10}x^{10} + a_9x^9 + \dots + a_1x + a_0 = (((...(a_{10}x + a_9)x + a_8)x + \dots + a_1)x + a_0.$$

§ 25

Организация ввода и вывода данных с использованием файлов

До сих пор мы программировали ввод исходных данных с клавиатуры и вывод на экран монитора. Как было сказано в § 17, вводом/выводом называется обмен данными между оперативной памятью и любыми внешними устройствами, в том числе устройствами внешней памяти: магнитными и оптическими дисками, флеш-накопителями.

Информация на внешних носителях хранится в файлах. По форме хранения данных файлы бывают *типизированными*, *нетипизированными* и *текстовыми*. В типизированных и нетипизированных файлах данные различных типов хранятся в том же формате, что и в оперативной памяти. Следовательно, при чтении и записи в такие файлы данные копируются, не изменяя объема и формы своего представления. В текстовых файлах данные хранятся в символьном формате. Поэтому при вводе (чтении) чисел из текстового файла происходит преобразование их представления из символьной формы в форму их внутреннего представления (с фиксированной или плавающей запятой). А при выводе (записи) чисел в текстовый файл они преобразуются из внутренней формы в символьную.

Далее мы будем использовать только текстовые файлы.

Текстовые файлы

Текстовый файл — наиболее часто употребляемая разновидность файлов. Устройства ввода с клавиатуры и вывода на экран работают только с текстовыми файлами. Файлы, содержащие тексты программ на Паскале и других языках программирования, являются текстовыми. Различная документация, информация, передаваемая по каналам электронной связи, — всё это текстовые файлы.

Содержимое текстового файла представляет собой символьную последовательность, разделенную на строки. Каждая строка заканчивается специальным признаком EOLN (*end of line* — конец строки). Весь файл заканчивается признаком EOF (*end of file* — конец файла). Схематически это выглядит так:

S_1	S_2	...	S_{k1}	EOLN	S_1	S_2	...	S_{k2}	EOLN	...	EOF
-------	-------	-----	----------	------	-------	-------	-----	----------	------	-----	-----

Здесь S_i обозначает i -й символ в строке. Каждый символ представлен во внутреннем коде (ASCII) и занимает 1 байт. Признак EOLN состоит из двух однобайтовых управляющих кодов: CR (код ASCII — 13) — возврат к началу строки и LF (код ASCII — 10) — перевод строки. При выводе содержимого текстового файла на экран или на печать признак EOLN обеспечивает визуальное разделение строк: переход к продолжению вывода с новой строки.

Текстовый файл можно создать или преобразовать с помощью текстового редактора. Его можно просмотреть на экране монитора или распечатать на принтере.

Ввод из текстового файла

Исходные данные могут быть заранее подготовлены в файле с помощью текстового редактора и сохранены на диске под определенным именем. Ввод исходных данных из файла производится автоматически, и при этом не происходит задержки выполнения программы, как при клавиатурном вводе.

Для организации ввода данных из текстового файла следует:

- объявить в программе переменную с типом Text (она называется файловой переменной);
- связать файловую переменную с файлом внешней памяти, содержащим исходные данные, с помощью оператора Assign;
- открыть файл для чтения с помощью процедуры Reset;

- осуществить чтение из файла с помощью операторов `Read` или `ReadLn`;
- закрыть файл с помощью оператора `Close`.

Пример 1. В текстовом файле с именем `abc.txt` хранятся пять чисел, разделенных на две строки:

```
2.5  3.1  4.0
0.7  1.5
```

В следующей программе организован ввод этих данных в вещественные переменные *a*, *b*, *c*, *d*, *e*.

```
Var a, b, c, d, e: Real;
      FD: Text; {Описание файловой переменной}
Begin
  Assign(FD, 'abc.txt'); {переменная FD связывается
                        с файлом abc.txt}
  Reset(FD); {файл открывается для чтения с его начала}
  ReadLn(FD, a, b, c); {чтение первой строки файла}
  ReadLn(FD, d, e); {чтение второй строки файла}
  Close(FD); {разрывается связь переменной FD с файлом}
  ...
```

Здесь `FD` — файловая переменная. `Assign`, `Reset`, `ReadLn`, `Close` — операторы обращения к стандартным процедурам, имеющим следующие форматы:

```
Assign(<файловая переменная>, <имя файла>)
Reset(<файловая переменная>)
ReadLn(<файловая переменная>, <список ввода>)
Close(<файловая переменная>)
```

Если файл хранится не в текущем каталоге, то в операторе `Assign` кроме имени файла надо указывать полный путь к нему. Имя файла можно задавать в строковой константе или переменной.

Для массивов большого размера удобно производить ввод значений из заранее подготовленного текстового файла с исходными данными.

Пример 2. Пусть в текстовом файле с именем `matr.txt` с помощью текстового редактора записана следующая числовая матрица размером 4×4 :

5	7	10	3
3	2	1	23
7	12	6	10
9	2	6	14

В следующей программе производится ввод этой матрицы в двумерный массив *M*:

```
Var M: array[1..4,1..4] of Integer;
    i, j: Byte;
    F1: Text; {файловая переменная}
begin
    Assign(F1, 'matr.txt'); {Связывание F1
                             с файлом matr.txt}
    Reset(F1); {Открытие файла для чтения}
    For i:=1 To 4 Do
    Begin
        For j:=1 To 4 Do
            Read(F1, M[i,j]); {Последовательное чтение
                               из одной строки}
        ReadLn(F1) {Переход к следующей строке}
    End;
    Close(F1); {Закрытие файла}
    ...
```

Вывод в текстовый файл

Запись результатов выполнения программы в текстовый файл позволяет сохранить их для того, чтобы в дальнейшем можно было их просмотреть с помощью текстового редактора, распечатать на принтере, а также использовать в качестве исходных данных для другой программы.

Для организации вывода данных в текстовый файл следует:

- объявить в программе файловую переменную с типом `Text`;
- связать файловую переменную с файлом внешней памяти с помощью оператора `Assign`;
- открыть файл для записи с помощью процедуры `Rewrite`;

- осуществить запись в файл с помощью операторов Write или WriteLn;
- закрыть файл с помощью оператора Close.

Пример 3. Требуется записать в текстовый файл таблицу умножения на 2.

```
Var A: Integer;  
    TM: Text; {Описание файловой переменной}  
Begin  
    Assign(TM, 'E:\TabMul.txt'); {Связывание переменной  
                                TM с файлом}  
    Rewrite(TM); {Открытие файла для записи}  
    {Циклический вывод в файл таблицы умножения}  
    For A:=2 To 9 Do WriteLn(TM, 2, '*', A, '=', 2*A);  
    Close(TM) {Закрытие файла}  
End.
```

Процедуры открытия файла для записи и запись в файл имеют следующий формат:

```
Rewrite(<файловая переменная>)  
Write(<файловая переменная>, <список вывода>)  
WriteLn(<файловая переменная>, <список вывода>)
```

Если файла с именем, указанным в операторе Assign, на диске не было, то программа его создаст. Если такой файл уже был, то его прежнее содержание будет утеряно и в него запишутся новые данные. В конце выполнения оператора WriteLn выставляется признак EOLN. Оператор Write этого не делает. Закрытие файла приводит к выставлению признака EOF.

В результате выполнения программы в корневом каталоге диска E появится файл с именем TabMul.txt. Открыв его в текстовом редакторе, увидим:

```
2*2=4  
2*3=6  
2*4=8  
2*5=10  
2*6=12  
2*7=14  
2*8=16  
2*9=18
```



Система основных понятий

Ввод и вывод в файлы
Текстовый файл хранит любую информацию в символьном формате, разделен на строки кодами EOLN, заканчивается кодом EOF
Ввод — передача данных с внешнего устройства в оперативную память
Ввод из файла: <code>Read(<ФП>, <список ввода>)</code> или <code>ReadLn(<ФП>, <список ввода>)</code> <ФП> — файловая переменная, для текстового файла объявляемая с типом <code>Text</code>
Вывод — передача данных из оперативной памяти на внешнее устройство
Вывод в файл: <code>Write(<ФП>, <список вывода>)</code> или <code>WriteLn(<ФП>, <список вывода>)</code> <ФП> — файловая переменная, для текстового файла объявляемая с типом <code>Text</code>
Операторы (стандартные процедуры) работы с файлами
<code>Assign</code> — назначение связи между файловой переменной и файлом на внешнем устройстве; <code>Reset</code> — открытие файла для чтения; <code>Rewrite</code> — открытие файла для записи; <code>Close</code> — закрытие файла (разрыв связи с файловой переменной)



Вопросы и задания

1. Сформулируйте правила организации ввода данных из текстового файла.
2. Сформулируйте правила организации вывода данных в текстовый файл.
3. Напишите программу, по которой из текстового файла с именем `kvur.txt` будут прочитаны три числа a , b , c — коэффициенты квадратного уравнения, затем будут вычислены корни этого уравнения и выведены на экран и в текстовый файл `korni.txt`.





4. Введите из текстового файла целочисленную матрицу размером 6×8 . Переверните матрицу, поменяв 1-ю строку с 6-й строкой, 2-ю строку с 5-й, 3-ю строку с 4-й и запишите полученную матрицу в другой файл.
5. Введите с клавиатуры одномерный числовой массив из 9 элементов. Сверните его в матрицу размером 3×3 , разместив первую тройку элементов в 1-й строке матрицы, 2-ю тройку — во второй строке, 3-ю тройку — в третьей строке. Полученную матрицу выведите на экран и в текстовый файл.
6. Введите с клавиатуры построчно в двумерный массив числовую матрицу размером 4×4 . Разверните ее по столбцам в одномерный массив. Запишите массив в текстовый файл.

§ 26

Типовые задачи обработки массивов

Заполнение массива

Значения массива могут задаваться вводом с клавиатуры, чтением из файла или вычислением в программе. В некоторых задачах статистического характера требуется заполнять массивы случайными числами.

Пример 1. Заполнить массив равномерно распределенными целыми случайными числами в диапазоне от 0 до 100.

```
Var X: array[1..20] of integer; i: integer;  
Begin  
  Randomize;  
  For i:=1 To 20 Do X[i]:=Random(100);  
  For i:=1 To 20 Do write(X[i]:4)  
End.
```

Со стандартной функцией `Random(x)` вы уже знакомы. Напомним, что она возвращает псевдослучайное целое число в диапазоне от 0 до $x - 1$.

Если требуется изменить диапазон случайных чисел, то это всегда можно сделать путем сдвига. Например, если нужно получить числа в диапазоне от -50 до 50 , то в программе пишется оператор присваивания:

```
X[i]:=Random(100)-50;
```

Для получения вещественных случайных чисел используется функция `Random` без аргумента. Она возвращает случайные дробные значения в диапазоне $[0, 1)$. С помощью сдвига и множителя эти значения можно привести к любому диапазону. Например, следующее выражение будет вычислять случайное вещественное число в диапазоне значений от -5 до 5 : $10 * \text{Random} - 5$.

Пример 2. Заполнить верхнетреугольную матрицу указанного вида и вывести ее на экран.

Матрица:	Программа решения задачи:
1 2 3 4	Var M: array [1..4,1..4] of Integer;
0 2 3 4	i, j: Integer;
0 0 3 4	Begin
0 0 0 4	For i:=1 To 4 Do
	For j:=1 To 4 Do
	If j<i Then M[i,j]:=0 Else M[i,j]:=j;
	For i:=1 To 4 Do
	Begin
	For j:=1 To 4 Do
	Write(M[i,j]:3);
	WriteLn
	End
	End.

Пояснение: для элементов $M[i, j]$ матрицы M , расположенных в верхнем треугольнике (включая диагональ), выполняется следующее соотношение между индексами: $j \leq i$.

Пример 3. Выбор максимального элемента. В одномерном массиве X из примера 1 требуется определить наибольшее значение среди значений элементов и его порядковый номер (индекс).

Идея алгоритма решения этой задачи следующая: чтобы в переменной $X_{\max}$ получить максимальное значение массива X , сначала в нее заносится первое значение массива $X[1]$. Затем значение $X_{\max}$ поочередно сравнивается с остальными элементами массива, и каждое значение, большее $X_{\max}$, присваивается этой переменной. Для получения номера максимального элемента массива в целочисленной переменной $i_{\max}$ следует записывать в нее номер элемента массива X одновременно с занесением значения в $X_{\max}$. На Алгоритмическом языке это запишется так:

```

Xmax:=X[1]; imax:=1
для I:=2 до 20
нц
    если X[I]>Xmax
    то
        Xmax:=X[I]; imax:=I
    все
кц

```

Заметим, что если в массиве *X* несколько значений, равных максимальному, то в *imax* будет получен номер первого из этих элементов. Чтобы получить номер последнего элемента, равного максимальному, нужно в ветвлении **если** заменить знак отношения *>* на *>=*. Для нахождения минимального элемента массива достаточно заменить знак отношения «больше» на «меньше».

Оформим в виде процедуры на Паскале подпрограмму поиска максимального элемента в одномерном массиве. Заполним одномерный массив случайными числами (как в примере 1). С помощью процедуры найдем в нем максимальное значение и индекс его первого вхождения в массив.

```

Const   n = 20;
Type    mas = array[1..n] of Integer;
Var     X: mas; i, Xmax, imax: Integer;
{Начало описания процедуры}
Procedure MaxArray(Var A:mas; Var MaxA, k:Integer);
Var     j: Integer;
Begin   MaxA:=A[1]; k:=1;
        For j:=2 To n Do
            If A[j]>MaxA Then
                Begin MaxA:=A[j]; k:=j End
        End; {Конец описания процедуры}
Begin
    Randomize;
    For i:=1 To n Do X[i]:=Random(100); {Заполнение
                                         массива}
    MaxArray(X, Xmax, imax); {Обращение к процедуре}
    Write('Xmax=', Xmax, 'imax=', imax) {Вывод
                                         результатов}
End.

```

Процедура *MaxArray* имеет три параметра: исходный массив *A*, *MaxA* — переменную для найденного максимального значения, *k* — переменную для индекса максимального значения. При обра-

щении к процедуре им соответствуют фактические параметры: X , $X_{\max}$, $i_{\max}$. Размер массива определяется глобальной константой n , значение которой используется как в основной программе, так и в процедуре.

Пример 4. Сортировка массива. В одномерном массиве X из N элементов требуется произвести перестановку значений так, чтобы они расположились по возрастанию, т. е. $X_1 \leq X_2 \leq \dots \leq X_N$.

Существует целый класс алгоритмов сортировки. Ниже описан алгоритм, который называется **методом пузырька**.

Идея алгоритма: производится последовательное упорядочивание смежных пар элементов массива: X_1 и X_2 , X_2 и X_3 , ..., X_{N-1} и X_N . В итоге максимальное значение переместится в X_N . Затем ту же процедуру повторяют до X_{N-1} и т. д., вплоть до цепочки из двух элементов X_1 и X_2 . Такой алгоритм будет иметь структуру двух вложенных циклов, причем внутренний цикл переменной (сокращающейся) длины.

```

для I:=1 до N-1
нц
  для J:=1 до N-I
нц
  если X[J]>X[J+1]
то
  A:=X[J];
  X[J]:=X[J+1];
  X[J+1]:=A
все
кц
кц

```

Для сортировки массива по убыванию значений достаточно заменить знак отношения «больше» на «меньше».

Запрограммируем на Паскале процедуру сортировки массива по возрастанию методом пузырька.

```

Const  n=20;
Type:  vector = array[1..n] of Real;
{описание процедуры сортировки}
Procedure SortArray(Var X: вектор);
Var i,j: Integer; A: Real;
Begin
  For i:=1 To n-1 Do
    For j:=1 To n-i Do

```

```

If X[j]>X[j+1] Then
  Begin A:=X[j]; X[j]:=X[j+1]; X[j+1]:=A End
End; {Конец описания процедуры}

```

Пример 5. В § 24 было рассмотрено описание двумерного массива, содержащего среднемесячные температуры за 10 лет, с 2001 по 2010 год. Определить, в каком году за этот период было самое теплое лето, т. е. в каком году была наибольшая средняя температура летних месяцев.

Идея решения: в одномерном массиве S получить средние температуры летних месяцев за каждый год из 10 лет. Затем найти номер наибольшего элемента в этом массиве, это и будет искомым год.

```

Program Example_2;
Type Month = array[1..12] of Real;
      Year = array[2001..2010] of Month;
Var Tabl: Year;
      S: array[2001..2010] of Real;
      I, J, K: Integer;
Begin {Ввод данных с клавиатуры}
  For I:=2001 To 2010 Do
    For J:=1 To 12 Do
      Begin
        Write(J:2, '.', I:4, ': ');
        ReadLn(Tabl[I, J])
      End;
  {Вычисление вектора средних летних температур}
  For I:=2001 To 2010 Do
    Begin
      S[I]:=0;
      For J:=6 To 8 Do
        S[I]:=S[I]+Tabl[I, J];
      S[I]:=S[I]/3
    End;
  {Определение года с самым теплым летом}
  K:=2001;
  For I:=2002 To 2010 Do
    If S[I]>S[K] Then K:=I;
  WriteLn('Самое теплое лето было в ',
        K, '-м году')
End.

```



Система основных понятий

Задачи обработки массива	
Типовые задачи: – заполнение массива вводом данных, вычислением значений, случайными числами; – поиск в массиве: заданного значения, максимального или минимального значения; – сортировка массива	
Датчик случайных, равномерно распределенных чисел	
Целые числа — случайные числа в диапазоне $[0, x-1]$: функция $\text{Random}(x)$, где x — целое число	Вещественные числа — случайные числа в диапазоне $(0, 1]$: функция Random (без аргумента)



Вопросы и задания

1. Какими способами можно заполнить массив значениями?
2. Как можно вычислять целые случайные числа в диапазоне от -50 до 0 ?
3. Как можно вычислять вещественные случайные числа в диапазоне от $2,5$ до 10 ?
4. Даны два вектора $\{x_i\}$, $\{y_i\}$, $i = 1, \dots, 10$, упорядоченные по возрастанию. Соедините их в один вектор $\{z_i\}$, $i = 1, \dots, 20$ так, чтобы сохранилась упорядоченность.
5. Дан массив, состоящий из 100 целых чисел. Выведите все числа, которые встречаются в этом массиве:
 - а) несколько раз;
 - б) только по одному разу.
6. В целочисленной матрице размером 10×10 найдите максимальное значение и индексы всех элементов, равных ему.
7. Матрицу размером 10×10 заполните случайными двоичными цифрами (0 и 1). Определите номер строки с наибольшим количеством нулей.
8. В двоичной матрице размером 10×10 (см. задание 7) найдите совпадающие строки.



§ 27

Символьный тип данных

Величина типа «символ» может принимать значения любых символов компьютерного алфавита. Символьная величина занимает 1 байт памяти, в котором хранится код этого символа, соответствующий используемой кодовой таблице. Заметим, что в Delphi наряду с однобайтовой кодировкой символов используется и двухбайтовая.

Символьная константа записывается между апострофами. Например: 'R', '+', '9', ']'.

Символьной тип называется Char. Пример описания символьных переменных:

```
Var c1, c2: Char;
```

Символьный тип относится к порядковым типам данных. Из этого следует:

- символы — упорядоченное множество;
- у каждого символа в этом множестве есть свой порядковый номер;
- между символами работает соотношение «следующий — предыдущий».

Порядковый номер символа — это его десятичный код, который лежит в диапазоне от 0 до 255. Например, в кодовой таблице ASCII десятичный код латинской буквы 'A' равен 65, а цифры '5' — 53. О стандартах кодирования символов рассказывалось в § 6.

Функция Ord (x)

Ord(x) — функция от аргумента порядкового типа, которая возвращает порядковый номер значения x в этом типе данных. Если x — символьная величина, то результатом функции будет десятичный код x в кодовой таблице. Например:

```
Ord('A')= 65, Ord('5')= 53
```

Функция Chr (x)

Chr(x) — функция от целочисленного аргумента, результатом которой является символ с кодом, равным x. Например:

```
Chr(65)='A', Chr(53)='5'
```

Поскольку коды символов лежат в диапазоне от 0 до 255, желательно тип *x* определять либо как `byte`, либо как интервальный тип `0..255`.

Пример 1. Составить программу на Паскале, по которой на экран будет выводиться таблица кодировки в диапазоне кодов от 32 до 255. Напомним, что символы с кодами, меньшими 32, являются управляющими (не экранными).

```

Program Tabl_code;
Var kod: Byte; {Целые числа от 0 до 255}
Begin
  For kod:=32 To 255 Do {Перебор кодов символов}
    Begin
      If (kod mod 10=0) Then WriteLn; {Перевод строки
                                     через 10 шагов}
      Write(chr(kod):3, kod:4);      {Вывод символа
                                     и его кода}

    End
End.

```

Значения выводятся парами: символ — код. В одной строке располагается 10 таких пар. Вся таблица разместится в 24 строках на экране.

Принцип последовательного кодирования алфавитов

В любой кодовой таблице выполняется принцип последовательного кодирования латинского (английского) алфавита и алфавита десятичной системы счисления. Это важное обстоятельство, которое часто учитывается в программах обработки символьной информации.

При выполнении операций отношений, применительно к символьным величинам, учитываются коды этих величин. Чем больше значение кода, тем символ считается больше. Истинными являются следующие отношения: 'A' < 'B', 'Z' > 'Y', 'a' > 'A'. Значение символьной переменной *C* является прописной (заглавной) латинской буквой, если истинно логическое выражение:

`(C>='A') and (C<='Z')`

Значение символьной переменной *C* является цифрой, если истинно логическое выражение:

```
(C>='0') and (C<='9')
```

В латинском алфавите 26 букв. Поэтому разница между кодами букв 'Z' и 'A', а также 'z' и 'a' равна 25.

Пример 2. С помощью датчика случайных чисел заполнить массив `Sim[0..10]` строчными английскими буквами. Затем массив отсортировать в алфавитном порядке.

```
Uses CRT;
Var Sim: array[0..10] of Char;
    C: Char; i, k: Integer;
Begin
  ClrScr;
  Randomize; {Заполнение массива случайными буквами}
  WriteLn('Исходный массив:');
  For i:= 0 To 10 Do
    Begin
      Sim[i]:=Chr(Random(26)+Ord('a'));
      Write(Sim[i])
    End;
  WriteLn;
  {Сортировка методом пузырька}
  For i:=0 To 9 Do
    For k:=0 To 9-i Do
      If Sim[k]>Sim[k+1] Then
        Begin
          C:=Sim[k]; Sim[k]:=Sim[k+1]; Sim[k+1]:=C
        End;
  WriteLn('Отсортированный массив:');
  For i:=0 To 10 Do Write(Sim[i]);
End.
```

При тестировании программы было получено:

Исходный массив:

gnkbeqgmsin

Отсортированный массив:

beggikmnngs



Система основных понятий

Символьный тип данных
Величины символьного типа (Char): константы и переменные, принимающие значения символов компьютерного алфавита
1 символ занимает 1 байт памяти (в 8-битовых кодировках)
Ord (x) — функция от аргумента порядкового типа, которая возвращает порядковый номер значения x в этом типе данных. Если x — символьная величина, то функция возвращает код символа
Chr (x) — функция от целочисленного аргумента, результатом которой является символ с кодом, равным x
В любой кодовой таблице выполняется принцип последовательного кодирования латинского (английского) алфавита и алфавита десятичной системы счисления



Вопросы и задания

1. Как в программе на Паскале обозначаются символьные константы и переменные?
2. С помощью какой стандартной функции определяется код символа?
3. С помощью какой стандартной функции можно определить символ по его коду?
4. Что такое принцип последовательного кодирования алфавитов? Приведите примеры алгоритмов, где он может быть использован.
5. Определите результаты вычисления выражений (типы и значения):
 - a) `Chr(Ord('B'))`
 - б) `Ord('A') - Ord('Z')`
 - в) `Ord('A') - Ord('a') = Ord('Z') - Ord('z')`
 - г) `Ord('9') - Ord('0')`
 - д) `Chr(Ord('a') + Ord('R') - Ord('r'))`
6. Выполните на компьютере программы из примеров 1 и 2 данного параграфа. Протестируйте их работу.



§ 28

Строки символов

Рассмотрим еще один структурный тип данных — строковый тип. Он позволяет программировать обработку слов, предложений, текстов.

Строка — это последовательность символов. Каждый символ занимает 1 байт памяти (код ASCII). Количество символов в строке называется ее длиной. Длина строки может находиться в диапазоне от 0 до 255. Строковые величины могут быть константами и переменными.

Строковая константа записывается как последовательность символов, заключенная в апострофы. Например:

```
' Язык программирования ПАСКАЛЬ '  
' IBM PC - computer '  
' 33-45-12 '
```

Строковая переменная описывается в разделе описания переменных следующим образом:

```
Var <идентификатор>: String[<максимальная длина  
строки>]
```

Например:

```
Var Name: String[20]
```

Параметр длины может и не указываться в описании. В таком случае подразумевается, что он равен максимальной величине — 255. Например:

```
Var slovo: String
```

Строковая переменная занимает в памяти на 1 байт больше, чем указанная в описании длина. Дело в том, что один (нулевой) байт содержит значение *текущей длины строки*. Если строковой переменной не присвоено никакого значения, то ее текущая длина равна нулю. По мере заполнения строки символами ее текущая длина возрастает, но она не должна превышать максимальной по описанию величины.

Символы внутри строки индексируются (нумеруются), начиная с единицы. Каждый отдельный символ идентифицируется именем строки с индексом, заключенным в квадратные скобки. Например:

```
Name[5], Name[i], slovo[k+1].
```

Значение индекса может быть задано положительной константой, переменной, выражением целочисленного типа. Оно не должно выходить за границы описания.

Тип *String* и стандартный тип *Char* совместимы: строки и символы могут употребляться в одних и тех же выражениях.

Строковые выражения строятся из строковых констант, переменных, функций и знаков операций. Над строковыми данными допустимы операция сцепления и операции отношения.

Операция сцепления (+) применяется для соединения нескольких строк в одну результирующую строку. Сцеплять можно как строковые константы, так и переменные.

Например:

```
'ЭВМ'+' IBM'+' PC'
```

В результате получится строка:

```
'ЭВМ IBM PC'
```

Длина результирующей строки не должна превышать 255.

Операции отношения: =, <, >, <=, >=, <> производят сравнение двух строк, в результате чего получается логическая величина (*true* или *false*). Операции отношения имеют более низкий приоритет, чем операция сцепления. Сравнение строк производится слева направо до первого несовпадающего символа, и та строка считается больше, в которой первый несовпадающий символ имеет больший номер в таблице символьной кодировки.

Если строки имеют различную длину, но в общей части символы совпадают, считается, что более короткая строка меньше, чем более длинная. Строки равны, если они полностью совпадают по длине и содержат одни и те же символы.

Пример

Выражение:	Результат:
'cosm1' < 'cosm2'	True
'pascal' > 'PASCAL'	True
'Ключ_' <> 'Ключ'	True
'MS DOS' = 'MS DOS'	True

Функции и процедуры

Функция `Copy(S, Poz, N)` выделяет из строки *S* подстроку длиной *N* символов, начиная с позиции *Poz*. *N* и *Poz* — целочисленные выражения.

Пример

Значение <i>S</i> :	Выражение:	Результат:
'ABCDEFGH'	<code>Copy(S, 2, 3)</code>	'BCD'
'ABCDEFGH'	<code>Copy(S, 4, 4)</code>	'DEFG'

Функция `Concat(S1, S2, ..., SN)` выполняет сцепление (конкатенацию) строк *S1*, ..., *SN* в одну строку.

Пример

Выражение:	Результат:
<code>Concat('AA', 'XX', 'Y')</code>	'AAXXY'

Функция `Length(S)` определяет текущую длину строки *S*. Результат — значение целочисленного типа.

Пример

Значение <i>S</i> :	Выражение:	Результат:
'test-5'	<code>Length(S)</code>	6
'(A+B)*C'	<code>Length(S)</code>	7

Функция `Pos(S1, S2)` обнаруживает первое появление в строке *S2* подстроки *S1*. Результат — целое число, равное номеру позиции, где находится первый символ подстроки *S1*. Если в *S2* не обнаружена подстрока *S1*, то результат равен 0.

Пример

Значение <i>S2</i> :	Выражение:	Результат:
'abcdef'	Pos('cd', S2)	3
'abcdcdef'	Pos('cd', S2)	3
'abcdef'	Pos('k', S2)	0

Процедура Delete(*S*, *Poz*, *N*) удаляет *N* символов из строки *S*, начиная с позиции *Poz*.

Пример

Исходное значение:	Оператор:	Конечное значение:
'abcdefg'	Delete(<i>S</i> , 3, 2)	'abefg'
'abcdefg'	Delete(<i>S</i> , 2, 6)	'a'

В результате выполнения процедуры уменьшается текущая длина строки в переменной *S*.

Процедура Insert(*S1*, *S2*, *Poz*) выполняет вставку строки *S1* в строку *S2*, начиная с позиции *Poz*.

Пример

Начальное <i>S2</i> :	Оператор:	Конечное <i>S2</i> :
'ЭВМ PC'	Insert('IBM-', S2, 5)	'ЭВМ IBM-PC'
'Рис. 2'	Insert('N', S2, 6)	'Рис. N2'

Примеры программ обработки строк

Пример 1. Составить программу, формирующую символьную строку, состоящую из *N* звездочек (*N* — целое число, $1 \leq N \leq 255$).

```

Program Stars;
Var A: String;
      N, I: Byte;
Begin
  Write('Введите число звездочек');
  ReadLn(N);
  A:='';
  For I:=1 To N Do
    A:=A+'*';
  WriteLn(A)
End.

```

Здесь строковой переменной *A* вначале присваивается значение пустой строки, обозначаемой двумя апострофами (''). Затем к ней присоединяются звездочки.

Пример 2. В символьной строке подсчитать количество цифр, предшествующих первому символу '!'.


```

Program C;
Var S: String;
      K, I: Byte;
Begin
  WriteLn('Введите строку');
  ReadLn(S);
  K:=0; I:=1;
  While (I<=Length(S)) And (S[I]<>'!') Do
    Begin
      If (S[I]>='0') And (S[i]<='9')
      Then K:=K+1;
      I:=I+1
    End;
  WriteLn('Количество цифр до символа "!" равно ', K)
End.

```

В этой программе переменная *K* играет роль счетчика цифр, а переменная *I* — роль параметра цикла. Цикл закончит выполнение при первом же выходе на символ '!' или, если в строке такого символа нет, то при выходе на конец строки. Символ *S[I]* является цифрой, если истинно отношение: $'0' \leq S[I] \leq '9'$.

Система основных понятий

Строки символов
Строка — последовательность символов
Описание строковой переменной: Var <идентификатор>: String[<длина строки>] Максимальная длина строки — 255
Обозначение символа в строке: <идентификатор строки>[<индекс>]
Операции над строками: сцепление (+), отношение (=, <, >, <=, >=, <>)
Стандартные функции: Copy(S, Poz, N) — выделение подстроки; Concat(S1, S2, ..., SN) — сцепление (конкатенация) строк; Length(S) — определение текущей длины строки; Pos(S1, S2) — определение первого вхождения подстроки в строку
Стандартные процедуры: Delete(S, Poz, N) — удаление подстроки; Insert(S1, S2, Poz) — вставка подстроки



Вопросы и задания

1. Как в программе обозначается строковая константа, как определяется строковая переменная?
2. Какой может быть максимальная длина строки?
3. Составьте программу получения из слова «дисквод» слова «воск», используя операцию сцепления и функцию `Copy`.
4. Составьте программу получения слова «правило» из слова «операция», используя процедуры `Delete`, `Insert`.
5. В данном слове замените первый и последний символы на символ '*'.
 В данном слове произведите обмен первого и последнего символов.
7. К данному слову присоедините столько символов '!', сколько в нем имеется букв (например, из строки 'УРА' надо получить 'УРА!!!').
8. В данной строке вставьте пробел после каждого символа.
9. Переверните введенную строку (например, из 'ДИСК' должно получиться 'КСИД').
10. В данной строке удалите все пробелы.
11. Строка представляет собой запись целого числа. Составьте программу ее перевода в соответствующую величину целого типа.



§ 29

Комбинированный тип данных

Все структурные типы данных, с которыми вы уже познакомились (массивы, строки), представляют собой совокупности однотипных величин. **Комбинированный тип данных** — это структурный тип, состоящий из фиксированного числа компонентов (полей) разных типов.

Комбинированный тип объявляется в программе в разделе типов:

```

Type <имя> = record
    <имя поля 1>: <тип>;
    ...
    <имя поля N>: <тип>
End

```

Поля могут иметь любые типы, в том числе и комбинированный тип.

Например, данные о результатах экзаменов, полученных учеником по трем предметам, могут быть представлены одной величиной комбинированного типа:

```
Type results = record
  Family: string[15]; {Фамилия ученика}
  Rus: 2..5;           {Оценка по русскому языку}
  Alg: 2..5;           {Оценка по алгебре}
  Phiz: 2..5           {Оценка по физике}
End;
```

После этого в разделе переменных следует описание:

```
Var exam: results;
```

Величина комбинированного типа называется **записью**. Элементы записи идентифицируются составными именами следующей структуры:

<имя переменной>.<имя поля>

Например: exam.family, exam.rus

В программе может использоваться массив, элементами которого являются записи.

Пример 1. На экзаменационном листе содержатся сведения о результатах экзаменов, сданных 30 учениками класса. Ввести эти данные в компьютер и получить список всех отличников.

В программе используется описание комбинированного типа results, приведенное выше. Исходные данные организуются в массив следующей структуры.

```
Var list: array[1..30] of results;
```

После ввода в этот массив исходных данных следует фрагмент программы:

```
For i:=1 To 30 Do
  If (list[i].rus=5) and (list[i].alg=5)
    and (list[i].phiz=5)
  Then WriteLn(list[i].family);
```

Программа отбирает записи, в которых все поля с оценками равны 5, и выводит соответствующие поля фамилий.

А теперь обсудим проблему: как наиболее удобным способом организовать ввод данных в этой программе? Вводить с клавиатуры неудобно из-за большого объема данных. При каждом повторном запуске программы нужно начинать ввод сначала. А при отладке это наверняка придется делать многократно. Гораздо удобнее подготовить файл с исходными данными с помощью текстового редактора. После этого без проблем можно повторять ввод многократно. Так и поступим. Подготовим текстовый файл следующего вида:

Таблица успеваемости 10А класса			
Фамилия	Русский язык	Алгебра	Физика
Антонов	4	5	5
Андреева	5	3	4
Боброва	5	5	5
...			

Таблица содержит данные с фамилиями и оценками 30 учеников класса. Обратите внимание на то, что фамилии записываются в отдельных строках. Необходимость этого связана с реализацией алгоритма (см. далее): при вводе символьной строки прочитывается полностью очередная строка текстового файла до признака EOLN. При этом фамилии должны содержать не более 15 символов, а первые оценки (по русскому языку) — располагаться не раньше 16-й позиции в своей строке.

Сохраним этот файл в корневом каталоге логического диска E под именем 10_a.txt. Составим программу с вводом таблицы успеваемости и выводом списка отличников. Фамилии отличников выведем на экран и сохраним в файле с именем Best.txt.

```
Program Examen;  
Type results = record  
    Fam: string[15];  
    Rus: 2..5;  
    Alg: 2..5;  
    Phiz: 2..5  
End;
```

```
Var list: array[1..30] of results; {Массив записей}
  i: Integer; F1, F2: text;
Begin
  Assign(F1, 'E:\10_a.txt'); {Связывание F1 с файлом 10_a.txt}
  Assign(F2, 'E:\Best.txt'); {Связывание F2 с файлом Best.txt}
  Reset(F1);                  {Открытие файла F1 для чтения}
  Rewrite(F2);                 {Открытие файла F2 для записи}
  ReadLn(F1); ReadLn(F1); {Пропуск 2 строк в файле F1}
  {Цикл ввода из файла F1}
  For i:=1 To 30 Do
    ReadLn(F1, list[i].Fam, list[i].Rus, list[i].Alg,
      list[i].Phiz);
  {Цикл отбора отличников и вывода их фамилий}
  For i:=1 To 30 Do
    If (list[i].rus=5) and (list[i].alg=5)
      and (list[i].phiz=5)
    Then
      Begin
        WriteLn(list[i].fam); {Вывод фамилии на экран}
        WriteLn(F2, list[i].fam) {Запись фамилии в файл F2}
      End;
    Close(F1); Close(F2)      {Закрытие файлов}
End.
```

Пример 2. Решая рассмотренную задачу с оценками, можно обойтись без массива записей. Кроме того, можно не ставить ограничения на число учеников в классе. Их число выяснится в процессе чтения файла с таблицей успеваемости. Составим программу, которая кроме вывода списка фамилий отличников подсчитает их количество и процент отличников по отношению к полному составу класса.

```
Program Examen_2;
Type results = record
  Fam: string[15];
  Rus: 2..5;
  Alg: 2..5;
  Phiz: 2..5
End;
Var list: results; {Одна переменная комбинированного
  типа}
  I, K: Integer;
  F1, F2: text;
```

Begin

```

Assign(F1, 'E:\10_a.txt');
Assign(F2, 'E:\Best.txt');
Reset(F1); Rewrite(F2);
ReadLn(F1); ReadLn(F1);
I:=0; K:=0; {Инициализация счетчиков}
{Цикл до конца чтения файла}
While not EOF(F1) Do
  Begin
    ReadLn(F1, list.Fam, list.Rus, list.Alg, list.Phiz);
    I:=I+1; {Подсчет числа учеников}
    If (list.rus=5) and (list.alg=5) and (list.phiz=5)
      Then
        Begin
          WriteLn(list.fam);
          WriteLn(F2, list.fam);
          K:=K+1; {Подсчет числа отличников}
        End
      End;
    WriteLn('Из ', I, ' учеников в классе ', K,
      ' отличников, что составляет ', K/I*100:4:1, '%');
    Close(F1); Close(F2) {Закрытие файлов}
  End.

```

В этой программе переменная *I* используется как счетчик числа учеников, а переменная *K* — как счетчик числа отличников.

Стандартная логическая функция EOF (end of file) примет значение True, когда процесс чтения из файла дойдет до его конца.

В результате выполнения программы кроме списка отличников на экран выведется строка:

Из 30 учеников в классе 10 отличников, что составляет 33,3%



Система основных понятий

Комбинированный тип данных
Комбинированный тип данных — структурный тип, объединяющий разнотипные компоненты (поля) данных
Тип поля: любой простой или структурированный тип (кроме файлового)
Запись — величина комбинированного типа
Идентификация поля записи — составное имя: <имя записи>.<имя поля>

Вопросы и задания



1. Чем комбинированный тип данных отличается от регулярного типа данных (массива)?
2. Что такое запись?
3. Опишите комбинированный тип для записей, содержащих следующие данные учеников: фамилию, имя, год рождения, рост (в сантиметрах), вес (в килограммах).
4. Опишите содержимое текстового файла, из которого будут вводиться данные, соответствующие описанию из предыдущего задания, для нескольких учеников класса (не менее пяти).
5. Напишите программу, по которой будут введены данные из файла, описанного в предыдущем задании, и выполнена следующая обработка:
 - определение среднего роста и среднего веса всех учеников;
 - вывод на экран и в файл `rost.txt` списка (фамилии, имена, возраст) учеников, рост которых выше среднего роста;
 - вывод на экран и в файл `ves.txt` списка учеников, вес которых меньше среднего веса.
6. Решите предыдущую задачу, не используя в программе массива записей. *Подсказка:* оператор `Reset` можно использовать в программе многократно для повторного чтения файла, начиная с его первой записи.



ЭОР к главе 3 на сайте ФЦИОР (<http://fcior.edu.ru>)



- Алгоритмы сортировки
- Вложенные циклы (на примере языка Pascal)
- Использование цикла While-do (на примере языка Pascal). (Практическая работа)
- Конструирование логических выражений
- Начальные сведения о программах на языке Pascal
- Объявление переменных в программе (на примере языка Pascal). Использование. Присваивание. Практическая работа
- Объявление переменных в программе. Перечислимые и интервальные типы (на примере языка Pascal). Практическая работа
- Операторы ветвления `if` и `case` (на примере языка Pascal). Практическая работа
- Основные структуры данных

- Основные типы данных: Integer, Real, Boolean, Character и String. Работа с переменными и константами (на примере языка Pascal) (И)
- Основные элементы языка программирования (на примере языка Pascal). Циклы. Работа с циклами. Использование циклов в программе. Вложенные циклы
- Основы работы со строками в языке Pascal. Практическая работа
- Основы составления программы, осуществляющей вывод данных на консоль в языке Pascal
- Простейшие операции языка Pascal
- Работа с массивами. Одномерные массивы. Алгоритмы работы с массивами. Обработка массива в цикле. Подсчет суммы элементов, максимум и минимум, поиск и сортировка элементов в массиве (на примере языка Pascal). (И)
- Реализация основных алгоритмических конструкций. Создание шаблона программы на языке Pascal
- Функции работы со строками в языке Pascal. Практическая работа
- Этапы разработки программы, ее структура. Создание шаблона программы на языке Pascal

Ответы на вопросы и задания

Глава 1

§ 3. 8. 600 бит.

10. 1 СИМВОЛ.

§ 4. 8. а) $i = \log_2 6 = 2,58$; б) $i = \log_2 12 = 3,58$;

в) $i = \log_2 7 = 2,81$; г) $i = \log_2 30 = 4,91$.

9. 11 бит.

§ 5. 3. 01111111.

4. 10000001.

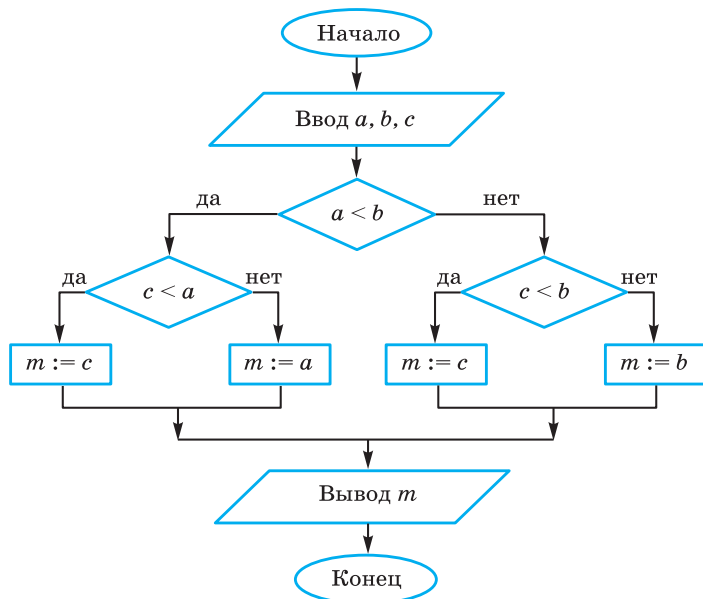
Глава 2

§ 8. 7. $\approx 7,63 \cdot 10^{-6}$ с.

8. $\approx 9,16 \cdot 10^{-6}$ с.

Глава 3

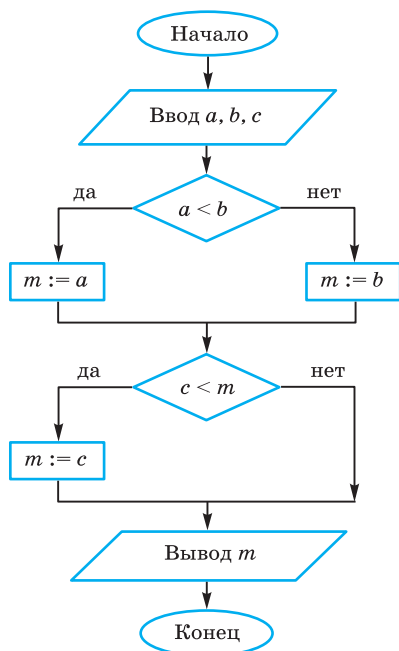
§ 13. 4. Вариант 1:



```

алг минимум трех чисел (с вложенными ветвлениями)
вещ  $a, b, c, min$ 
нач ввод  $a, b, c$ 
    если  $a < b$ 
    то
        если  $c < a$ 
        то  $m := c$ 
        иначе  $m := a$ 
    все
    иначе
        если  $c < b$ 
        то  $m := c$ 
        иначе  $m := b$ 
    все
все
вывод  $m$ 
кон
    
```

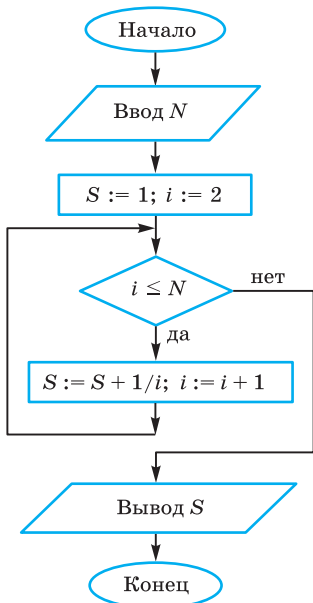
Вариант 2:



```

алг минимум трех чисел
    (с последовательными
    ветвлениями)
вещ  $a, b, c, m$ 
нач ввод  $a, b, c$ 
    если  $a < b$ 
    то  $m := a$ 
    иначе  $m := b$ 
все
если  $c < m$ 
то  $m := c$ 
все
вывод  $m$ 
кон
    
```

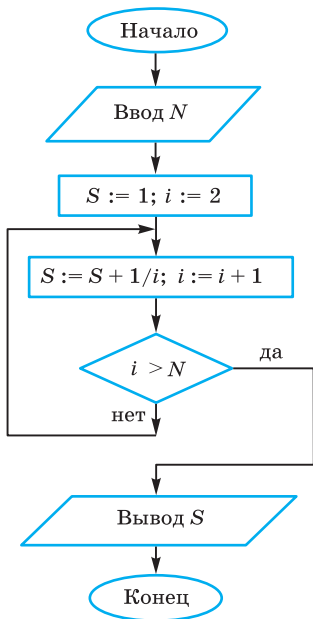
5. Вариант 1:



```

алг сумма ряда (цикл-пока)
цел N, i
вещ S
нач ввод N
  S:=1
  i:=2
  пока i<=N
  нц
    S:=S+1/i
    i:=i+1
  кц
  вывод S
кон
    
```

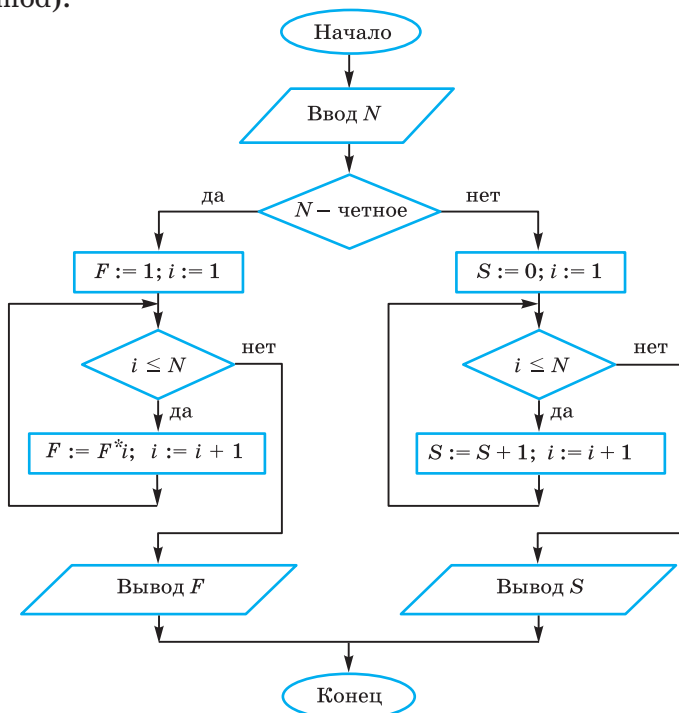
Вариант 2:



```

алг сумма ряда (цикл-до)
цел N, i
вещ S
нач ввод N
  S:=1
  i:=2
  повторить
    S:=S+1/i
    i:=i+1
  до i>N
  вывод S
кон
    
```

6. *Примечание.* В описании алгоритма на АЯ используется операция взятия остатка от целочисленного деления (mod).



```

алг ряд
цел N, i, S
нач ввод N
  i:=1
  если N mod 2=0
    то S:=1
    иначе S:=0
  все
  пока i <= N
  нц
    если N mod 2=0
      то S:=S*i
      иначе S:=S+i
    все
    i:=i+1
  кц
  вывод S
кон
    
```

§ 17. 3. Правильные: а, г, д. Неправильные: б, в, е, ж, з.

```
4. Program zadanie_17_4;
   Var x, y: integer;
   Begin
       Writeln('Введите 2 целых числа: x и y');
       Read(x,y);
       x:=x+y;
       y:=x-y;
       x:=x-y;
       Write('После обмена значениями x=',x,' y=',y);
   End.
```

При использовании вещественных чисел может дать неточный результат.

5. $h:=k \text{ div } 100 \text{ mod } 10$

```
6. Program zadanie_17_6;
   Var k,s: integer;
   Begin
       Read(k);
       s:=k mod 10 *1000 + k div 10 mod 10 *100 +
           k div 100 mod 10*10 + k div 1000;
       Write(s);
   End.
```

```
7. Program zadanie_17_7;
   Var n1, n2: integer;
   Begin
       Read(n1);
       n2:=n1 mod 10 + n1 div 10 mod 10*2 +
           n1 div 100 mod 10*4 + n1 div 1000*8;
       Write(n2);
   End.
```

§ 18. 3. а) $(X = Y) \& (X = Z) \& (Y = Z)$; б) $(X = Y) \vee (X = Z) \vee (Y = Z)$;
 в) $(X > 0) \& (Y > 0) \& (Z > 0)$; г) $((X > 0) \& (Y < 0) \& (Z < 0)) \vee$
 $\vee ((X < 0) \& (Y > 0) \& (Z < 0)) \vee ((X < 0) \& (Y < 0) \& (Z > 0))$;
 д) $(X < Y) \& (Y < Z)$.

4. а) $(X=Y) \text{ and } (X=Z) \text{ and } (Y=Z)$; б) $(X=Y) \text{ or } (X=Z) \text{ or } (Y=Z)$;
 в) $(X>0) \text{ and } (Y>0) \text{ and } (Z>0)$; г) $((X>0) \text{ and } (Y<0) \text{ and } (Z<0)) \text{ or } ((X<0) \text{ and } (Y>0) \text{ and } (Z<0)) \text{ or } ((X<0) \text{ and } (Y<0) \text{ and } (Z>0))$; д) $(X<Y) \text{ and } (Y<Z)$.

5.

X	Y	Z	$\neg X \& Y \vee X \& Z$
И	И	И	И
И	И	Л	Л
И	Л	И	И
И	Л	Л	Л
Л	И	И	И
Л	И	Л	И
Л	Л	И	Л
Л	Л	Л	Л

6. а) false; б) true; в) true; г) false; д) false; е) false.

§ 21.1.

Номер шага	N	E	i	a	комментарий
1	3	0	0	1	До входа в цикл
2	3	0	0	1	$i \leq N$ (т. е. $0 \leq 3$) да
3	3	1	1	1	
4	3	1	1	1	$i \leq N$ (т. е. $1 \leq 3$) да
5	3	2	2	0,5	
6	3	2	2	0,5	$i \leq N$ (т. е. $2 \leq 3$) да
7	3	2,5	3	0,167	
8	3	2,5	3	0,167	$i \leq N$ (т. е. $3 \leq 3$) да
9	3	2,667	4	0,042	
10	3	2,667	4	0,042	$i \leq N$ (т. е. $4 \leq 3$) нет
11		2,667			Результат

Примечание. Вещественные числа в таблице округлены до трех цифр после запятой.

```

3. Program zadanie_21_3;
Var x, k1, k2, i: integer;
Begin
  k1:=0;
  k2:=0;
  For i:=1 To 10 Do
  Begin
    Read(x); If odd(x)
              Then k2:=k2+1
              Else k1:=k1+1;
  End;
  Write('Четных - ', k1);
  Write(' Нечетных - ', k2);
End.

```

```

4. Program zadanie_21_4;
   Var i,j: integer;
   Begin
     For i:=1 To 10 Do
       Begin
         For j:=1 To 10 Do Writeln(i,'*',j,'=',i*j);
         Writeln;
       End;
     End.

```

```

7. Program zadanie_21_7;
   Var x: real;
   Begin
     x:=0;
     Writeln('  x          sin x          cos x');
     Writeln('-----');
     While x<=1 Do
       Begin
         Writeln('  ',x:6:4,'  ',sin(x):6:4,'  ',
                 cos(x):6:4);
         x:=x+0.1;
       End
     End.

```

Примечание. Для выравнивания значений в каждом столбце в программе используется форматный вывод. Запись x:6:4 означает, что под число отводится 6 разрядов, из которых 4 будет выделено под дробную часть.

```

8. Program zadanie_21_8;
   Var i: integer;
   Begin
     For i:=100 To 999 Do
       If (i mod 10<>i div 10 mod 10) and (i mod 10<>
         i div 100) and (i div 100<>i div 10 mod 10)
       Then Writeln(i);
     End.

```

```

§ 22. 1. Program zadanie_22_1;
   Var x,y,z:integer;
   Begin
     for x:=1 to 10 do
       for y:=1 to 10 do
         for z:=1 to 10 do
           if sqr(x)+sqr(y)=sqr(z) then
             writeln(x,'^2+',y,'^2=',z,'^2')
         end;
       end;
     end.

```

```

7. Program zadanie_22_7;
   var X: Longint; chet, nechet: integer;
   Begin

```

```

Write('Введите целое число: '); ReadLn(X);
chet:=0; nechet:=0;
While (X>0) Do
  Begin
    if X mod 10 mod 2=0
      then chet:=chet+1
      else nechet:=nechet+1;
    X:=X div 10;
  End;
WriteLn('Четных цифр = ', chet);
WriteLn('Нечетных цифр=', nechet);
End.

```

§ 23. 4. Реализация с процедурой:

```

Program zadanie_23_4;
var R1,R2,S1,S2: real;
Procedure square_krug(x:real; Var square:real);
Begin
  square:=PI*sqr(x);
End;
Begin
  Write('Введите внешний радиус: '); ReadLn(R1);
  square_krug(R1,S1);
  Write('Введите внутренний радиус: '); ReadLn(R2);
  square_krug(R2,S2);
  Write('Площадь круга равна ',S1-S2);
End.

```

Реализация с функцией:

```

Program zadanie_23_4;
var R1,R2: real;
Function square(x:real):real;
Begin
  square:=PI*sqr(x);
End;
Begin
  Write('Введите внешний радиус: '); ReadLn(R1);
  Write('Введите внутренний радиус: '); ReadLn(R2);
  Write('Площадь круга равна ',
    square(R1)-square(R2));
End.

```

```

7. Program zadanie_23_7;
var x,y,z,s1,s2,s3,m,chislo: integer;
Function summa(a:integer):integer;
Var S:integer;
Begin
  S:=0;
  While a>0 Do

```

```

    Begin S:=S+a mod 10; a:=a div 10; End;
    summa:=S;
End;
Begin
    Write('Введите три целых числа: '); ReadLn(x,y,z);
    s1:=summa(x); s2:=summa(y); s3:=summa(z);
    m:=s1; chislo:=x;
    if s2>m then Begin m:=s2; chislo:=y; End;
    if s3>m then Begin m:=s3; chislo:=z; End;
    Writeln('Наибольшая сумма цифр у числа ',chislo)
End.

```

§ 24. 5. Program zadanie_24_5;
Var z:array[1..50] **of** integer;
 i:integer; L:real;

```

Begin
    Writeln('Введите 50 целых чисел');
    For i:=1 to 50 Do ReadLn(z[i]);
    L:=0;
    For i:=1 to 50 Do L:=L+sqr(z[i]);
    L:=sqrt(L);
    Writeln('Длина вектора=',L)
End.

```

§ 25. 3. Примечание. Один из возможных вариантов выполнения задания. Рекомендуем сравнить эту реализацию алгоритма с той, что приведена в § 20.

```

Program zadanie_25_3;
Var a,b,c,d,x,x1,x2: real; F,F1: Text;
Begin
    Assign(F, 'D:\kvur.txt'); Reset(F);
    ReadLn(F,a,b,c); Close(F);
    If (a=0) and (b=0) and (c=0) Then
        Writeln('Любое x - решение');
    If (a=0) and (b=0) and (c<>0) Then
        Writeln('Нет решений');
    If (a=0) and (b<>0) Then
        Begin x:=-c/b; Writeln('x=', x) End;
    If a<>0 Then
        Begin d:=b*b-4*a*c;
            If d<0
                Then Writeln('Нет вещественных корней')
                Else
                    Begin
                        x1:=(-b+sqrt(d))/2/a;
                        x2:=(-b-sqrt(d))/2/a;
                        Writeln('x1=', x1); Writeln('x2=', x2);
                        Assign(F1, 'D:\korni.txt'); Rewrite(F1);
                        Writeln(F1, x1, x2); Close(F1)
                    End
                End
        End
    End
End.

```

Для того чтобы эта программа работала, необходимо вначале сформировать файл, содержащий коэффициенты:

```
Program coef_a_b_c;
Var a,b,c: real; F: text;
Begin
  Assign(F, 'D:\kvur.txt'); Rewrite(F);
  WriteLn('Введите коэффициенты a,b,c');
  WriteLn('a='); ReadLn(a); WriteLn(F,a);
  WriteLn('b='); ReadLn(b); WriteLn(F,b);
  WriteLn('c='); ReadLn(c); WriteLn(F,c);
  Rewrite(F); Write(F,a,b,c); Close(F)
End.
```

4. Program zadanie_25_4;

```
Var a: array[1..6,1..8] of integer;
    i,j,k: integer;
    f1,f2: text;
Begin
  Assign(f1,'matr1.txt'); Assign(f2,'matr2.txt');
  Reset(f1); Rewrite(f2);
  For i:=1 To 6 Do
    For j:=1 To 8 Do
      Read(f1,a[i,j]);
  For i:=1 To 3 Do
    For j:=1 To 8 Do
      Begin
        k:=a[i,j]; a[i,j]:=a[6-i+1,j]; a[6-i+1,j]:=k;
      End;
  For i:=1 To 6 Do
    Begin
      For j:=1 To 8 Do
        Write(f2,a[i,j], ' '); Writeln(f2);
    End;
  Close(f1); Close(f2);
End.
```

§ 26. 4. Program zadanie_26_4;

```
Var x,y:array[1..10] of integer;
    z:array[1..20] of integer;
    i,k1,k2:integer;
Begin
  Writeln('Введите первый вектор');
  For i:=1 to 10 Do Read(x[i]);
  Writeln('Введите второй вектор');
  For i:=1 to 10 Do Read(y[i]);
  k1:=1; //счетчик первого массива
  k2:=1; //счетчик второго массива
  i:=1;
```

```

While (k1<=10) and (k2<=10) Do //пока не закончился
                                //один из массивов
Begin
  if (x[k1]<y[k2]) Then //сравниваем текущие
                        //элементы двух массивов
    Begin
      z[i]:=x[k1]; //если текущий элемент
                  //первого массива меньше
      k1:=k1+1      //заносим его в массив z
    End
  Else
    Begin
      z[i]:=y[k2]; //если текущий элемент
                  //второго массива меньше
      k2:=k2+1      //заносим его в массив z
    End;
    i:=i+1;
  End;
While (k1<=10) Do //если в 1-м массиве остались
                  //элементы, то переносим их в z
  Begin z[i]:=x[k1]; i:=i+1; k1:=k1+1 End;
While (k2<=10) Do //если во 2-м массиве остались
                  //элементы, то переносим их в z
  Begin z[i]:=y[k2]; i:=i+1; k2:=k2+1 End;
For i:=1 to 20 Do Writeln(z[i]);
End.

7. Program zadanie_26_7;
Var x:array[1..10,1..10] of integer;
    z:array[1..10] of integer; //массив для хранения
                                //количества нулей в строках
    i,j,kol,max,n:integer;
Begin
  Randomize;
  For i:=1 to 10 Do //формирование матрицы
    For j:=1 to 10 Do
      x[i,j]:=Random(2);
    For i:=1 to 10 Do Begin //вывод сформированного
                            //массива
      For j:=1 to 10 Do
        Write(x[i,j]:3);
      Writeln;
    End;
  max:=0;
  For i:=1 to 10 Do Begin
    kol:=0;
    For j:=1 to 10 Do //подсчет нулей в i-й строке
      if x[i,j]=0 then kol:=kol+1;
  
```

```

    z[i]:=kol; //занесение количества нулей в массив z
    if kol>max Then max:=kol; //определение
                                //максимального количества нулей
End;
For i:=1 to 10 Do                //ищем строки с максимальным
                                //количеством нулей
    if z[i]=max Then
        writeln('Строка ',i);
End.

```

- § 27. 5. а) тип char, значение 'В'; б) тип integer, значение (-25);
 в) тип boolean, значение true; г) тип integer, значение 9;
 д) тип char, значение 'А'.

§ 28. 7. **Program** zadanie_28_7;
Var s:string; n,i:integer;
Begin
 ReadLn(s); n:=length(s);
 For i:=1 to n Do s:=s+'!';
 Writeln(s);
End.

10. **Program** zadanie_28_10;
Var s:string; n:integer;
Begin
 ReadLn(s);
 While Pos(' ',s)<>0 Do
 Begin n:=Pos(' ',s); Delete(s,n,1); End;
 Writeln(s);
End.

11. **Program** zadanie_28_11;
Var s:string;
 n,i,itog,cifra:integer;
Begin
 ReadLn(s); n:=length(s);
 itog:=0;
 For i:=1 to n Do
 Begin
 cifra:=ord(s[i])-ord('0'); //перевод символьного
 //представления цифры в число
 If i=1 Then itog:=cifra //если это первая цифра
 Else
 itog:=itog*10+cifra; //сдвиг разрядов влево
 //и добавление новой цифры в конец
 End;
 Writeln(itog);
End.



Практические работы к главе 1 «Информация»

Работа 1.1. Шифрование данных

Цель работы: знакомство с простейшими приемами шифрования и дешифрования текстовой информации.

Задание 1

Шифр Цезаря. Этот шифр реализует следующее преобразование текста: каждая буква исходного текста заменяется следующей после нее буквой в алфавите, который считается написанным по кругу.

Используя шифр Цезаря, зашифровать следующие фразы:

- а) Делу время — потехе час
- б) С Новым годом
- в) Первое сентября

Задание 2

Используя шифр Цезаря, декодировать следующие фразы:

- а) Лмбттоьк шбт
- б) Вёмпё тпмочё рфтуьой

Задание 3

Шифр Виженера. Это шифр Цезаря с переменной величиной сдвига. Величину сдвига задают ключевым словом. Например, ключевое слово ВАЗА означает следующую последовательность сдвигов букв исходного текста: 3 1 9 1 3 1 9 1 и т. д. Используя в качестве ключевого слово ЗИМА, закодировать слова:

АЛГОРИТМИЗАЦИЯ, КОМПЬЮТЕР, ИНТЕРНЕТ.

Задание 4

Слово ЖПЮЩЕБ получено с помощью шифра Виженера с ключевым словом БАНК. Восстановить исходное слово.

Задание 5*

С помощью табличного процессора Microsoft Excel автоматизировать процесс кодирования слов с использованием ключевого слова bank (предполагается, что слова будут состоять только из

¹⁾ Задания со «звездочкой» имеют повышенную сложность.

строчных латинских букв и их длина не будет превышать 10 символов). Для решения задачи использовать текстовые функции СИМВОЛ и КОДСИМВОЛ. Каждая буква должна храниться в отдельной ячейке. Величина сдвига должна определяться автоматически (код буквы ключевого слова минус код буквы «а» плюс единица). Попробовать с помощью вашей таблицы зашифровать слова: algebra, geometry, english.

Задание 6

Используя в качестве ключа расположение букв на клавиатуре вашего компьютера, декодировать сообщение:

D ktce hjlbkfcм kjxrf?

D ktce jyf hjckf?

Задание 7

Используя в качестве ключа расположение букв на клавиатуре вашего компьютера, закодировать сообщение:

Москва — столица России

Задание 8

Шифр перестановки. Кодирование осуществляется перестановкой букв в слове по одному и тому же правилу. Восстановить слова и определить правило перестановки:

НИМАРЕЛ, ЛЕТОФЕН, НИЛКЙЕА, НОМОТИР, РАКДНАША.

Задание 9

Используя приведенный в задании 8 шифр перестановки, закодировать следующие слова:

ГОРИЗОНТ, ТЕЛЕВИЗОР, МАГНИТОФОН.

Задание 10

Определить правило шифрования и расшифровать слова:

КЭРНОЦЛИТКЭЛУОНПИЕЖДАИФЯ

УКРОГРЕОШЛАЕКВИСЧТЕВМО

Задание 11

Используя приведенный ниже ключ, расшифровать сообщения.

Ключ: РА ДЕ КИ МО НУ ЛЯ

Сообщения:

а) АКБМУНИЯДКУМВРЛ ИКСЯМТР

б) ТДЯДФМУУЫЙ АРЗГМВМА

Задание 12

С помощью ключа из задания 11 закодировать пословицы:

- а) Рыбак рыбака видит издалека
- б) Сделал дело — гуляй смело

Задание 13*

Придумать свой ключ шифрования и закодировать с помощью него сообщение:

Бит — это минимальная единица измерения информации

Работа 1.2. Измерение информации

Цель работы: практическое закрепление знаний о способах измерения информации при использовании содержательного и объемного подходов.

Задание 1

Определить (с помощью утилиты «Установка и удаление программ»), сколько приложений установлено на вашем компьютере, и вычислить, какое количество информации содержит сообщение о том, что было запущено одно из них.

Задание 2

Врач-стоматолог принимает пациентов с 8 утра до 12 часов дня. На каждого пациента отводится по 30 минут. Какое количество информации содержит сообщение о том, что Петя записался на прием в 11.30?

Задание 3

Известно, что сообщение учителя о том, что к доске пойдет Саша Орлов, содержит 5 битов информации. Сколько учеников в классе?

Задание 4

В корзине лежат 4 груши и 12 яблок. Какое количество информации содержит сообщение о том, что из корзины достали грушу?

Задание 5

В школьную команду по волейболу было отобрано некоторое количество учеников из 64 претендентов. Сколько учеников было отобрано, если сообщение о том, кто был выбран, содержит 72 бита информации?

Задание 6

Из папки DOCUM было удалено 13 файлов, и сообщение об этом содержит 91 бит информации. Сколько файлов осталось в папке?

Задание 7

В детской магнитной азбуке 32 буквы. Какое количество информации содержит сообщение о том, что из них было собрано слово «муравей»?

Задание 8

Какое количество информации получит второй игрок при игре в крестики-нолики на поле размером 8×8 клеток после первого хода первого игрока, играющего крестиками?

Задание 9

Необходимо отгадать слово, состоящее из 5 букв и записанное с помощью алфавита из 32 букв. Можно задавать вопросы, ответом на которые будет «Да» или «Нет». С помощью какого числа вопросов можно отгадать слово при оптимальной стратегии игры?

Задание 10

Открыть в редакторе Microsoft Word произвольный текстовый файл (без графических элементов), определить количество символов в этом файле, включая пробелы (команда **Сервис** → **Статистика** или **Рецензирование** → **Статистика**), и вычислить количество информации в этом файле.

Задание 11

Алфавит некоторого языка состоит из 64 символов. За сколько секунд можно передать текст из 2000 символов, если скорость передачи — 50 байтов в секунду?

Задание 12

В некотором алфавите записан текст из 300 символов, количество информации в тексте — 150 байтов. Сколько символов в алфавите?

Задание 13

Приведенное ниже сообщение содержит все буквы алфавита. Какое количество информации оно содержит?

Ƴ&≍ℓ⊆ℓ ƳΨ↗⊆∂∩⋈⋈ ⋈ ⋈ ΔΣΨ⋈

Задание 14

Имеется файл с текстом из 20 000 символов. При наборе текста использовался компьютерный алфавит. Текст необходимо скопировать на диск, на котором имеется свободная область памяти 20 Кбайт. Поместится ли текст на диск?

Задание 15

В течение 10 секунд было передано сообщение, количество информации в котором равно 5000 байтов. Каков размер алфавита, если скорость передачи — 800 символов в секунду?

Задание 16

Два сообщения содержат одинаковое количество символов. Количество информации в первом тексте в 2,5 раза больше, чем во втором. Сколько символов содержат алфавиты, если известно, что число символов в каждом алфавите не превышает 32 и на каждый символ приходится целое число битов?

Задание 17

Сколько битов информации содержит любое трехзначное восьмеричное число?

Задание 18

Создать с помощью табличного процессора Microsoft Excel таблицу для автоматического перевода чисел из битов в байты, килобайты, мегабайты и гигабайты. Перевести во все предложенные единицы измерения 1000 битов, 8192 бита, 20 500 битов, 16 394 бита, 100 200 битов.

Задание 19*

Найти неизвестные x и y , если верны соотношения 16^y Мбайт = 8^x бит и 2^x Кбайт = 2^y Мбайт.

Задание 20*

Создать с помощью табличного процессора Excel таблицу следующего вида:

Из какой единицы измерения переводить?	Количество информации	Биты	Байты	Килобайты	Мегабайты	Гигабайты

В первом столбце единица измерения должна выбираться из списка (список создается с помощью команды **Данные** → **Проверка**). Далее, в зависимости от выбранной единицы измерения, заданное количество информации должно переводиться во все указанные единицы измерения (использовать условную функцию).

Протестировать работоспособность созданной таблицы и доказать правильность ее работы.

Работа 1.3. Представление чисел

Цель работы: закрепление знаний о системах счисления и о представлении чисел в памяти компьютера, полученных при изучении курса информатики основной школы.

Задание 1

Выписать алфавиты 2-ичной, 5-ричной, 8-ричной, 16-ричной систем счисления.

Задание 2

Записать первые 20 чисел натурального числового ряда в 2-ичной, 5-ричной, 8-ричной, 16-ричной системах счисления.

Задание 3

В какой системе счисления справедливо равенство:

а) $2 \cdot 2 = 10$; б) $2 \cdot 3 = 11$; в) $3 \cdot 3 = 13$?

Задание 4

Записать в развернутом виде числа.

а) $A_{10} = 125,34$; $A_8 = 125,34$; б) $A_6 = 125,34$; $A_{16} = 125,34$.

Пояснение. Развернутой формой записи числа называется запись вида:

$$A_q = \pm (a_{n-1}q^{n-1} + a_{n-2}q^{n-2} + \dots + a_0q^0 + a_{-1}q^{-1} + a_{-2}q^{-2} + \dots + a_{-m}q^{-m}).$$

Здесь A_q — число, q — основание системы счисления, a_i — цифры данной системы счисления, n — количество разрядов целой части числа, m — количество разрядов дробной части числа.

Например:

$$26,387_{10} = 2 \cdot 10^1 + 6 \cdot 10^0 + 3 \cdot 10^{-1} + 8 \cdot 10^{-2} + 7 \cdot 10^{-3};$$

$$101,11_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}.$$

В последнем примере использована десятичная развернутая форма записи двоичного числа.

Задание 5

Перевести числа в десятичную систему счисления.

- а) $A_8 = 341$; б) $A_6 = 341$; в) $A_{16} = 341$;
г) $A_5 = 34,1$; д) $A_{16} = E41A,12$.

Задание 6

Перевести целые числа из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную системы:

- а) 856; б) 664; в) 5012; г) 6435; д) 78.

Задание 7

Перевести десятичные дроби в двоичную и восьмеричную системы счисления, оставив пять знаков в дробной части нового числа.

- а) 21,5; б) 432,54; в) 678,333.

Задание 8

Составить таблицы сложения и умножения в двоичной системе счисления и выполнить вычисления:

- а) $1110 + 101$; б) $10101 - 11$; в) $101 \cdot 11$; г) $1110 : 10$.

Задание 9

Представить числа в двоичном виде в восьмибитовом представлении в формате целого без знака.

- а) 5; б) 17; в) 64; г) 255.

Задание 10

Представить числа в двоичном виде в восьмибитовом представлении в формате целого со знаком.

- а) 56; б) -56; в) 127; г) -127.

Задание 11*

Представить вещественные числа в четырехбайтовом представлении в формате с плавающей запятой.

- а) 0,5; б) 25,12; в) -25,12; г) -3456,1.

Работа 1.4. Представление текстов. Сжатие текстов

Цель работы: практическое закрепление знаний о представлении в компьютере текстовых данных.

Задание 1

Определить, какие символы кодировочной таблицы ASCII (DOS) соответствуют всем прописным буквам русского алфавита в кодировочной таблице ANSI (Windows). Для выполнения задания создать текст с русским алфавитом в текстовом редакторе «Блокнот», а затем открыть его в режиме просмотра (клавиша F3) в любом файловом менеджере (Windows Commander, Far, Total Commander, Norton Commander) и преобразовать в другую кодировку. После выполнения задания заполнить таблицу.

Буква в ANSI	Буква в ASCII	Буква в ANSI	Буква в ASCII	Буква в ANSI	Буква в ASCII
А		К		Х	
Б		Л		Ц	
В		М		Ч	
Г		Н		Ш	
Д		О		Щ	
Е		П		Ъ	
Ё		Р		Ы	
Ж		С		Ь	
З		Т		Э	
И		У		Ю	
Й		Ф		Я	

Задание 2

Закодировать текст с помощью кодировочной таблицы ASCII.

Happy Birthday to you!

Записать двоичное и шестнадцатеричное представления кода (для записи шестнадцатеричного кода использовать средство для просмотра файлов любого файлового менеджера).

Задание 3

Декодировать текст, записанный в международной кодировочной таблице ASCII (дано десятичное представление).

71 101 108 108 111 44 32 109 121 32 102 114 105 101 110
100 33

Задание 4

Пользуясь таблицей кодировки ASCII, расшифровать текст, представленный в виде двоичных кодов символов.

01010000 01100101 01110010 01101101 00100000 01010101
01101110 01101001 01110110 01100101 01110010 01110011
01101001 01110100 01111001

Задание 5

Пользуясь кодовой страницей Windows-1251 таблицы кодировки ASCII, получить шестнадцатеричный код слова ИНФОРМАТИЗАЦИЯ.

Задание 6

Во сколько раз увеличится объем памяти, необходимый для хранения текста, если его преобразовать из кодировки KOI8-R в кодировку Unicode?

Задание 7

С помощью табличного процессора Excel построить кодировочную таблицу ASCII, в которой символы будут автоматически отображаться на экране в соответствии с их заданным десятичным номером (использовать соответствующую текстовую функцию).

Работа 1.5. Представление изображения и звука

Цель работы: практическое закрепление знаний о представлении в компьютере графических данных и звука.

Справочная информация

В некоторых заданиях используется модельный (учебный) вариант монитора с размером раstra 10×10 пикселей.

При векторном подходе изображение рассматривается как совокупность простых элементов: прямых линий, дуг, окружностей, эллипсов, прямоугольников, закрасок и пр., которые называют-

ся *графическими примитивами*. Графическая информация — это данные, однозначно определяющие все графические примитивы, составляющие рисунок.

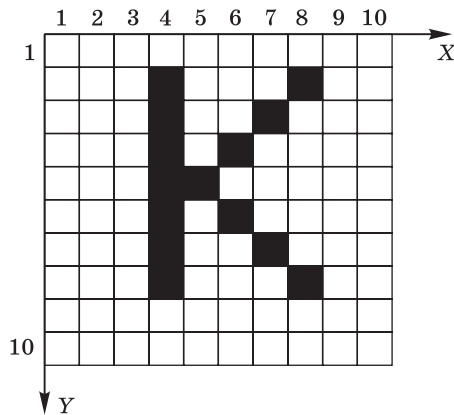
Положение и форма графических примитивов задаются в *системе графических координат*, связанных с экраном. Обычно начало координат расположено в верхнем левом углу экрана. Сетка пикселей совпадает с координатной сеткой. Горизонтальная ось X направлена слева направо; вертикальная ось Y — сверху вниз.

Отрезок прямой линии однозначно определяется указанием координат его концов; окружность — координатами центра и радиусом; многоугольник — координатами его углов, закрашенная область — граничной линией и цветом закрашки и пр.

Учебная система векторных команд представлена в таблице.

Установить X, Y	Установить текущую позицию (X, Y)
Линия к $X1, Y1$	Нарисовать линию от текущей позиции в позицию ($X1, Y1$), позиция ($X1, Y1$) становится текущей
Линия $X1, Y1, X2, Y2$	Нарисовать линию с координатами начала $X1, Y1$ и координатами конца $X2, Y2$. Текущая позиция не устанавливается
Окружность X, Y, R	Нарисовать окружность; X, Y — координаты центра, R — длина радиуса в пикселях
Эллипс $X1, Y1, X2, Y2$	Нарисовать эллипс, ограниченный прямоугольником; ($X1, Y1$) — координаты левого верхнего, а ($X2, Y2$) — правого нижнего угла этого прямоугольника
Прямоугольник $X1, Y1, X2, Y2$	Нарисовать прямоугольник; ($X1, Y1$) — координаты левого верхнего угла, а ($X2, Y2$) — правого нижнего угла этого прямоугольника
Цвет_рисования ЦВЕТ	Установить текущий цвет рисования
Цвет_закраски ЦВЕТ	Установить текущий цвет закрашки
Закрасить X, Y , ЦВЕТ ГРАНИЦЫ	Закрасить произвольную <i>замкнутую</i> фигуру; X, Y — координаты любой точки внутри замкнутой фигуры, ЦВЕТ ГРАНИЦЫ — цвет граничной линии

Например, требуется написать последовательность получения изображения буквы К:



Изображение буквы «К» на рисунке описывается тремя векторными командами:

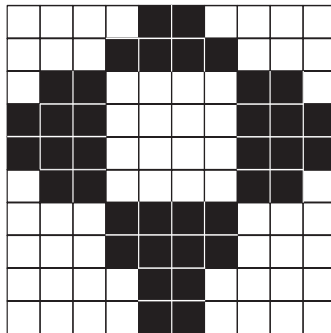
Линия(4, 2, 4, 8)

Линия(5, 5, 8, 2)

Линия(5, 5, 8, 8)

Задание 1

Построить двоичный код приведенного черно-белого растрового изображения, полученного на мониторе с размером растра 10×10 .



Задание 2

Определить, какой объем памяти требуется для хранения 1 бита изображения на вашем компьютере (для этого нужно через **Свойства экрана** определить битовую глубину цвета).

Задание 3

Битовая глубина цвета равна 24. Сколько различных оттенков серого цвета может быть отображено на экране (серый цвет получается, если уровни яркости всех трех базовых цветов одинаковы)?

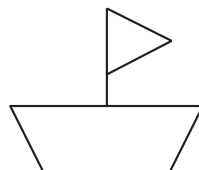
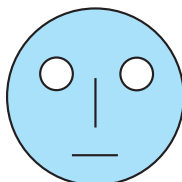
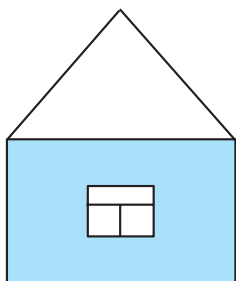
Задание 4

Дан двоичный код 8-цветного изображения. Размер монитора — 10×10 пикселей. Что изображено на рисунке (зарисовать)?

```
001 111 111 111 010 010 111 111 111 001
111 111 111 011 011 011 011 111 111 111
111 111 011 111 111 111 111 011 111 111
111 011 111 111 111 111 111 111 011 111
110 011 111 111 110 110 111 111 011 110
110 011 111 111 110 110 111 111 011 110
111 011 111 111 111 111 111 111 011 111
111 111 011 111 111 111 111 011 111 111
111 111 111 011 011 011 011 111 111 111
001 111 111 111 010 010 111 111 111 001
```

Задание 5

Описать с помощью векторных команд следующие рисунки (цвет заливки произвольный).



Задание 6

Получить растровое и векторное представления всех цифр от 0 до 9.

Задание 7

По приведенному ниже набору векторных команд определить, что изображено на рисунке (зарисовать).

Цвет рисования Голубой
 Прямоугольник 12, 2, 18, 8
 Прямоугольник 10, 1, 20, 21
 Прямоугольник 20, 6, 50, 21
 Цвет рисования Желтый
 Цвет закраски Зеленый
 Окружность 20, 24, 3
 Окружность 40, 24, 3
 Закрасить 20, 24, Желтый
 Закрасить 40, 24, Желтый
 Цвет закраски Голубой
 Закрасить 30, 10, Голубой
 Закрасить 15, 15, Голубой
 Цвет закраски Розовый
 Закрасить 16, 6, Голубой

Задание 8

Определить, какой объем имеет 1 страница видеопамати на вашем компьютере (узнать для этого, какое у компьютера разрешение и битовая глубина цвета). Ответ записать в мегабайтах.

Задание 9

Нарисовать в редакторе Paint изображение солнца, сохранить его в формате BMP, а затем с помощью Photoshop преобразовать его в форматы JPEG (с наивысшим качеством), JPEG (с наименьшим качеством), GIF, TIFF. Сравнить эффективность сжатия каждого формата, заполнив таблицу.

Формат	Размер файла	Коэффициент сжатия (по сравнению с BMP)
JPEG (высшее качество)		
JPEG (низкое качество)		
GIF		
TIFF		

Задание 10

Битовая глубина цвета равна 32. Видеопамять делится на две страницы. Разрешающая способность монитора 800×600 . Вычислить объем видеопамяти.

Задание 11

На компьютере установлена видеокарта объемом 2 мегабайта. Какое максимально возможное количество цветов теоретически допустимо в палитре при работе с монитором, имеющим разрешение 1280×1024 ?

Задание 12

Какой объем видеопамяти в килобайтах нужен для хранения изображения размером 600×350 пикселей, использующего 8-цветную палитру?

Задание 13

Зеленый цвет на компьютере с объемом страницы видеопамяти 125 Кбайт кодируется кодом 0010. Какова может быть разрешающая способность монитора?

Задание 14

Монитор работает с 16-цветной палитрой в режиме 640×400 пикселей. Для кодирования изображения требуется 1250 Кбайт. Сколько страниц видеопамяти оно занимает?

Задание 15

Сколько цветов можно максимально использовать для хранения изображения размером 350×200 пикселей, если объем страницы видеопамяти — 65 Кбайт?

Задание 16

Определить объем памяти для хранения цифрового аудиофайла, время звучания которого 5 минут при частоте дискретизации 44,1 кГц и глубине кодирования 16 битов.

Задание 17

Записать с помощью стандартного приложения «Звукозапись» звук длительностью 1 минута с частотой дискретизации

22,050 кГц и глубиной кодирования 8 битов (моно), а затем тот же самый звук с частотой дискретизации 44,1 кГц и глубиной кодирования 16 битов (моно). Сравнить объемы полученных файлов.

Задание 18

Одна минута записи цифрового аудиофайла занимает на диске 1,3 Мбайт, разрядность звуковой платы — 8. С какой частотой дискретизации записан звук?

Задание 19

Две минуты записи цифрового аудиофайла занимают на диске 5,1 Мбайт. Частота дискретизации — 22 050 Гц. Какова разрядность аудиоадаптера?

Задание 20

Объем свободной памяти на диске — 0,01 Гбайт, разрядность звуковой платы — 16. Какова будет длительность звучания цифрового аудиофайла, если его записать с частотой дискретизации 44 100 Гц?

Практические работы к главе 2 «Информационные процессы»

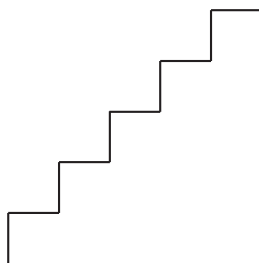
Работа 2.1. Управление алгоритмическим исполнителем

Цель работы: закрепление навыков программного управления учебными исполнителями алгоритмов, полученных при изучении курса информатики в 7–9 классах.

Используемое программное обеспечение: среда какого-либо учебного исполнителя алгоритмов графического типа, назначение которого — рисование на экране компьютера. К числу таких исполнителей относятся: Черепашка Лого, Чертежник, Кенгуренок и др.

Задание 1

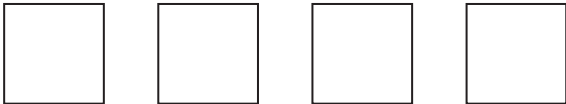
Написать подпрограмму (процедуру) STEP и с ее помощью составить программу рисования лесенки по диагонали через всё поле рисунка.



Задание 2

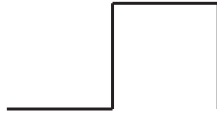
Написать программы для рисования следующих рисунков на всю ширину поля, используя вспомогательные алгоритмы (подпрограммы).

а) _____

б) 

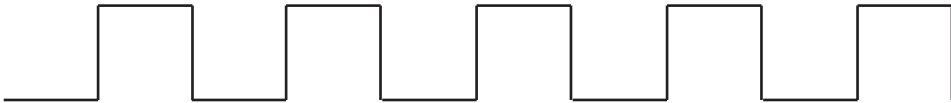
Задание 3

Описать подпрограмму для рисования следующей фигуры.



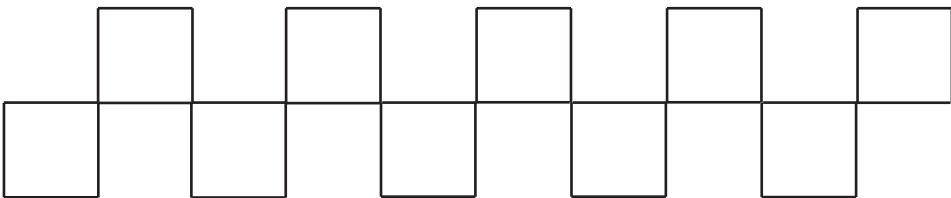
Задание 4

Используя подпрограмму из предыдущего задания, составить программу для рисования «забора» через всё поле рисунка.



Задание 5

Оформить решение задания 4 в виде подпрограммы и с ее помощью составить программу рисования следующей фигуры.



Работа 2.2. Автоматическая обработка данных

Цель работы: знакомство с основами теории алгоритмов на примере решения задач на программное управление алгоритмической машиной Поста.

Используемое программное обеспечение: имитатор машины Поста, который можно найти в Интернете по адресу

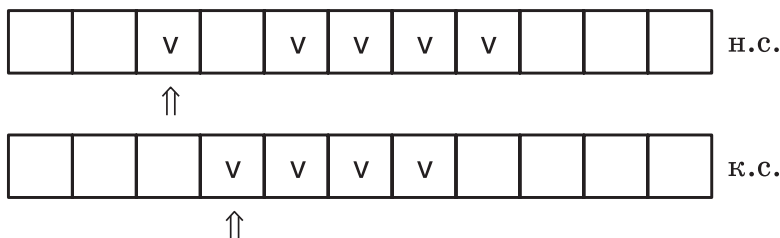
<http://priklinfa.narod.ru/anti800.htm>

Система команд машины Поста: (везде буква n обозначает номер текущей команды):

Команда	Действие
$n \leftarrow m$	Сдвиг каретки на шаг влево и переход к выполнению команды с номером m
$n \rightarrow m$	Сдвиг каретки на шаг вправо и переход к выполнению команды с номером m
$n \vee m$	Установка метки в текущую пустую клетку
$n \updownarrow m$	Стирание метки в текущей клетке
$n !$	Остановка выполнения программы
$n ? m, k$	Переход по содержимому текущей клетки: если текущая клетка пустая, то следующей будет выполняться команда с номером m ; если в текущей клетке стоит метка, то выполнится команда с номером k

Задание 1

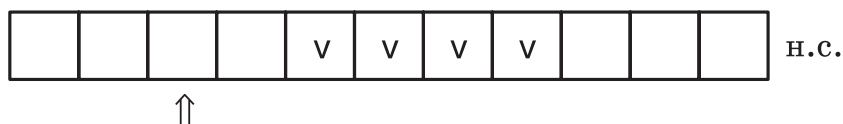
Составить программу перевода информационной ленты машины Поста из начального состояния (н.с.) в конечное (к.с.):



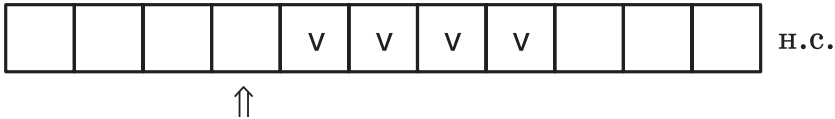
Задание 2

1. Выполнить на машине Поста программу:

1 $\vee$ 2	4 $\leftarrow$ 5
2 $\rightarrow$ 3	5 $\vee$ 6
3 ? 2,4	6 !



2. Какую задачу решает исполнитель по этой программе?
3. Что произойдет, если начальное состояние информационной ленты будет иметь следующий вид?



В следующих задачах считается, что n расположенных подряд меток обозначают число n (непозиционная система счисления с основанием 1).

Задание 3

Написать для машины Поста программу сложения двух чисел, записанных на ленте и расположенных через одну пустую клетку друг от друга. Начальное положение каретки — под пустой клеткой, отделяющей числа.

Задание 4

Написать для машины Поста программу вычитания двух чисел, разделенных одной пустой клеткой. Уменьшаемое не меньше вычитаемого. Начальное положение каретки — под пустой клеткой, отделяющей уменьшаемое от вычитаемого.

Указание. Стирать метки по одной у каждого числа, пока у вычитаемого не кончатся все метки.

Задание 5

Используя программу вычитания, проверить, что получится, если:

- а) уменьшаемое равно вычитаемому;
- б) уменьшаемое меньше вычитаемого.

Задание 6

Написать для машины Поста программу деления числа, записанного метками, на 2. Исходное число должно делиться на 2 без остатка.

Указание. Стереть каждую вторую метку; уплотнить оставшиеся метки.

Задание 7

Используя программу деления числа на 2:

- проверить, что получится для числа 2;
- модифицировать программу с учетом числа 2.

Указание. Справа от пустой клетки поставить метку, а слева стереть две метки. Так поступать до тех пор, пока слева остаются метки.

Задание 8

На информационной ленте машины Поста на расстоянии в n клеток друг от друга расположены две помеченные метками клетки. Начальное положение каретки — под левой из помеченных клеток. Какую работу выполнит Машина Поста по программе?

$1 \rightarrow 2$	$3 \leftarrow 4$
$2 ? 1,3$	$4 ? 3,1$

Задание 9

Написать для машины Поста программу умножения на 2 числа, записанного метками на ленте.

Указание. Через одну пустую клетку поставить две метки, а в исходном числе стереть одну. Так поступать, пока в исходном числе остаются метки.

Задание 10*

Написать для машины Поста программу, проверяющую, делится ли записанное метками число на 5.

Задание 11*

На информационной ленте машины Поста помечена $2n - 1$ клетка. Составить программу отыскания средней помеченной клетки и стирания метки в ней.

Задание 12*

На информационной ленте машины Поста расположены два массива помеченных клеток. Написать программу стирания меток, расположенных в большем массиве.

Работа 2.3. Проектное задание. Настройка BIOS

Цель работы: знакомство с процедурой первоначальной загрузки компьютера; получение представления о назначении BIOS; знакомство с основными приемами настройки BIOS, со средствами тестирования компьютера.

Справочная информация

Назначение BIOS. Прежде операционной системы в компьютере запускается встроенная в чип материнской платы программа **BIOS** (*Base Input/Output System*, основная система ввода-вывода). Назначение этого небольшого программного кода — поиск, тестирование и установка некоторых параметров устройств компьютера. Правильный подбор параметров может ускорить загрузку компьютера. BIOS представляет собой программу, записанную в микросхему ПЗУ по той или иной технологии ROM и, следовательно, не требующую питания для того, чтобы храниться там даже после выключения компьютера. Параметры настройки BIOS хранятся в энергозависимой CMOS RAM, которая питается от батарейки на материнской плате.

После включения питания напряжение подается на центральный процессор и другие микросхемы материнской платы. «Пробудившись», CPU запускает из микросхемы программу BIOS и начинается процедура POST (*Power On Self Test*, инициализация при первом включении). Ее задача — протестировать и настроить все «железо». Прежде всего формируется логическая архитектура компьютера. Подается питание на все чипсеты, в их регистрах устанавливаются нужные значения. Затем определяется объем ОЗУ (этот процесс можно наблюдать на экране), включается клавиатура, распознаются основные порты компьютера. На следующем этапе определяются блочные устройства — жесткие диски и приводы CD/DVD. На заключительной стадии происходит отображение итоговой информации. После окончания работы POST BIOS ищет загрузочную запись. Эта запись, в зависимости от настройки, находится на одном из жестких дисков, CD/DVD-ROM или устройстве USB. После того как загрузочная запись найдена, она загружается в память и управление передается ей.

Настройка BIOS. Для изменения настроек BIOS используется программа **Setup**, для входа в которую обычно используются кла-

виши Del или F2 (это зависит от производителя и версии BIOS). В настоящее время среди разработчиков BIOS для персональных компьютеров наиболее известны три фирмы: American Megatrends Inc. (AMI), Intel и Phoenix Technologies (торговые марки — Award BIOS, Phoenix Award BIOS). Именно их продукция встречается на подавляющем большинстве материнских плат. Однако даже для двух последовательных моделей материнских плат одного и того же производителя меню BIOS могут в той или иной степени различаться, так как единого стандарта на его интерфейс не существует. Многие современные материнские платы имеют графический интерфейс для работы с настройками BIOS (программы Setup).

Рассмотрим некоторые настройки BIOS:

- раздел **Main** или **Standard CMOS Setup**. Здесь можно задать дату и время, а также параметры жестких дисков;
- в разделе **Advanced BIOS Features (BIOS Features Setup** или просто **Advanced**) приведены различные общие настройки, позволяющие включить или отключить определенные опции загрузки компьютера;
- раздел **Integrated Peripherals** отвечает за интерфейсы, интегрированные устройства и дополнительные системные функции;
- раздел **Power Management Setup** позволяет настроить все опции энергопотребления и питания;
- раздел **Hardware Monitor** позволяет узнать значения системных датчиков: температуру процессора или скорость вращения вентиляторов (об/мин);
- пункт **Load Setup Defaults** восстанавливает настройки BIOS по умолчанию и устраняет все изменения, которые вы могли внести. Данный пункт будет полезен, если ваши действия привели к каким-либо проблемам в системе.

Main. Здесь можно установить время и дату, а также параметры ваших жестких дисков и других установленных накопителей. Каждый раз при загрузке ПК, скорее всего, автоматически определяет накопители, установленные в системе. У большинства компьютеров на это уходит секунда или две, но если вы вручную внесете нужные параметры, то несколько ускорите время загрузки.

Advanced BIOS Features. В этом разделе находятся различные опции, так или иначе относящиеся к специфичным настройкам BIOS, CPU, кэша и подобного. Здесь можно встретить следующие параметры (в скобках указаны различные варианты названий):

- *CPU Internal Frequency*. Конструкция некоторых материнских плат позволяет указать здесь частоту процессора. Однако будьте осторожны — «разгон» процессора может привести к его повреждению;
- *Boot Up NumLock Status*. Автоматическое включение цифровой клавиатуры, полезно для индивидуальной настройки;
- *Quick Power On Self Test (Quick Mode)*. Ускоряет загрузку, пропуская некоторые тесты, в том числе тройную проверку ОЗУ;
- *Boot Sequence*. Последовательность просмотра дисков для поиска загрузочного сектора. Этот режим может быть представлен и другим способом — в виде списка из четырех устройств. Обычно первым загрузочным устройством удобно ставить диск С. Кроме жестких дисков современные системы могут загружаться с CD-ROM.

В некоторых версиях BIOS последние 3 пункта могут находиться в разделе **Boot**.

Integrated Peripherals. Как правило, в материнскую плату встроен ряд контроллеров периферийных устройств: контроллер IDE, контроллер последовательных и параллельных портов, клавиатуры и пр. Иногда возникает необходимость отключения некоторых неиспользуемых устройств (в том числе интегрированных). В этом разделе обычно можно встретить следующие пункты:

- *Onboard IDE-1 Controller* — первый контроллер IDE-дисков;
- *Onboard IDE-2 Controller* — второй контроллер IDE-дисков;
- *USB Keyboard support*. Как известно, шина USB поддерживается средствами ОС. Таким образом, до загрузки Windows клавиатура работать не должна. Режим USB Keyboard support позволяет BIOS самостоятельно, на этапе загрузки, обрабатывать события, поступающие от клавиатуры;
- *Onboard Serial Port 1/2*. Этот параметр позволяет отключить порты COM1 и COM2;
- *Onboard Parallel Port* — отключение порта LPT (принтера).

Тестирование компьютера. В состав операционной системы Windows входит целый ряд служебных программ и утилит, позволяющих определить конфигурацию компьютера, установить версии ОС и BIOS, а также выполнить тестирование отдельных устройств.

К их числу относятся:

- утилита **Система (Настройка → Панель управления)**;
- утилита **Сведения о системе (Программы → Стандартные → Служебные)**;
- утилита **Администрирование (Настройка → Панель управления)**.

Задание 1

1. Определить тип и версию BIOS на вашем рабочем компьютере.
2. Установить порядок загрузки компьютера: CD-ROM, C.
3. Установить параметр ускоренной загрузки компьютера (отмена тройного тестирования памяти).
4. Установить автоматическое включение цифровой клавиатуры при загрузке компьютера.
5. Отключить порты COM и LPT.

Задание 2

1. Изучить возможности служебных программ и утилит компьютера.
2. Пользуясь изученными утилитами, определить следующие характеристики компьютера:

Характеристика	Значение
Название видеокарты и объем видеопамати	
Количество жестких дисков, их объемы	
Название звуковой карты	
Название сетевой карты	
Размер файла подкачки	
Версия ОС	
Версия BIOS	

3. Понаблюдать за степенью загрузки процессора в различных состояниях и заполнить таблицу:

Действие	Процент загрузки процессора в момент выполнения действия
Закрыты все приложения	
Запуск приложения MS Word	
Запуск приложения Paint	
Переключение в Word	
Процесс набора произвольного текста	
Переключение в Paint	
Процесс рисования произвольного объекта	
Закрытие обоих приложений	

4. Проверить необходимость дефрагментации жесткого диска.

Задание 3*

1. Скачать из Интернета последнюю версию программы CPU-Z (для ее поиска воспользоваться любым поисковым сервером).
2. Запустить программу CPU-Z, с ее помощью определить характеристики процессора на вашем рабочем компьютере и заполнить таблицу:

Характеристика	Значение
Название процессора	
Ядро (core)	
Тип разъема (socket)	
Тактовая частота	
Технологические нормы (в микронах)	
Напряжение питания ядра	

Практические работы к главе 3 «Программирование обработки информации»

Работа 3.1. Программирование линейных алгоритмов

Задание

Для каждой вычислительной задачи составить программу, содержащую операторы ввода, вывода, присваивания.

Уровень 1

1. Вычислить длину окружности и площадь круга одного и того же заданного радиуса R .
2. Вычислить расстояние между двумя точками с данными координатами на плоскости (x_1, y_1) и (x_2, y_2) .
3. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
4. Три сопротивления R_1, R_2, R_3 соединены параллельно. Найти сопротивление всей цепи.
5. Найти сумму членов арифметической прогрессии, если известны ее первый член, разность и число членов прогрессии.
6. Вычислить корни квадратного уравнения $ax^2 + bx + c = 0$ с заданными коэффициентами a, b и c (предполагается, что $a \neq 0$ и что дискриминант уравнения неотрицателен).
7. Найти площадь равнобедренной трапеции с основаниями a и b и углом α при большем основании a .

Уровень 2

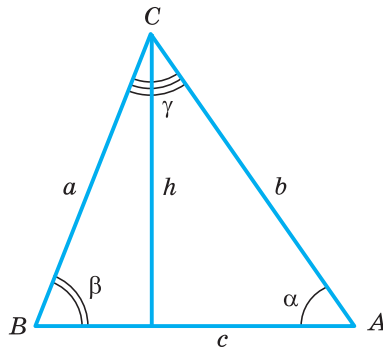
8. Заданы координаты трех вершин треугольника $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Найти его периметр и площадь.
9. Найти произведение всех цифр заданного четырехзначного числа.
10. Вычислить площадь и периметр правильного N -угольника, описанного около окружности радиуса R (рассмотреть N — целого типа, R — вещественного типа).

11. Дано натуральное число T — длительность прошедшего времени в секундах. Вывести данное значение длительности в часах (HH), минутах (MM) и секундах (SS) в следующей форме: HH ч MM мин SS с.
12. Дано действительное число R вида nnn.ddd (три цифровых разряда в целой и дробной частях). Поменять местами дробную и целую части числа и вывести полученное значение числа.
13. Составить программу перевода радианной меры угла в градусы, минуты и секунды.
14. С начала суток часовая стрелка повернулась на y градусов ($0 \leq y < 360$, y — вещественное число). Определить число полных часов и полных минут, прошедших с начала суток. Сформулировать и решить обратную задачу.

Уровень 3

15. Дан произвольный треугольник ABC , для которого определен следующий набор характерных параметров: a, b, c — стороны треугольника; α, β, γ — углы (в градусах); h — высота, опущенная на сторону c ; S — площадь; P — периметр треугольника. По трем заданным параметрам вычислить все остальные. Различные сочетания параметров определяют варианты заданий.

Замечание: входные (исходные) и выходные значения углов представить в градусной мере.



- | | | |
|----------------------|--------------------------|---------------------------|
| 1) a, b, c ; | 2) a, b, γ ; | 3) c, α, β ; |
| 4) h, c, b ; | 5) h, c, α ; | 6) S, h, b ; |
| 7) S, h, α ; | 8) a, b, h ; | 9) a, b, S ; |
| 10) a, b, P ; | 11) a, h, α ; | 12) a, h, γ ; |
| 13) S, c, α ; | 14) h, α, β ; | 15) h, α, γ . |

Работа 3.2. Программирование логических выражений

Задание

Для каждой задачи составить программу, выводящую значение TRUE, если указанное высказывание является истинным, и FALSE — в противном случае (использовать условный оператор нельзя).

Уровень 1

1. Треугольник со сторонами a , b , c является равносторонним.
2. Целое число N является четным двузначным числом.
3. Треугольник со сторонами a , b , c является равнобедренным.
4. Среди чисел a , b , c есть хотя бы одна пара взаимно противоположных чисел.
5. Данные числа x , y являются координатами точки, лежащей в первой координатной четверти.
6. Данные числа c и d являются соответственно квадратом и кубом числа a .
7. Заданное натуральное число N является двузначным и кратно K .

Уровень 2

8. (x_1, y_1) и (x_2, y_2) — координаты левой верхней и правой нижней вершин прямоугольника. Точка $A(x, y)$ принадлежит данному прямоугольнику.
9. Данное четырехзначное число читается одинаково слева направо и справа налево.
10. В заданном натуральном трехзначном числе N имеется четная цифра.
11. Сумма каких-либо двух цифр заданного трехзначного натурального числа N равна третьей цифре.
12. Сумма цифр заданного четырехзначного числа N превосходит произведение цифр этого же числа на 1.
13. Сумма двух последних цифр заданного трехзначного числа N меньше заданного K , а первая цифра больше 5.

Уровень 3

14. Целая и дробная части заданного вещественного числа одинаковы.
15. Заданы координаты трех точек плоскости. Эти точки не лежат на одной прямой.
16. Первые две цифры в дробной части заданного вещественного числа совпадают с записью целой части этого числа.
17. Точка с координатами (x, y) принадлежит части плоскости, лежащей между прямыми $y = m$, $y = n$ ($m < n$).
18. Среди первых трех цифр из дробной части положительного вещественного числа есть нуль.
19. Шахматный король за один ход может переместиться с одного заданного поля на другое (каждое поле задано двумя координатами — целыми числами от 1 до 8).
20. Шахматный ферзь за один ход может переместиться с одного заданного поля на другое (каждое поле задано двумя координатами — целыми числами от 1 до 8).

Работа 3.3. Программирование ветвящихся алгоритмов

Задание 1

Для каждой задачи составить программу с ветвящейся структурой, используя условный оператор IF.

Уровень 1

1. Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник. Если да, то прямоугольный ли он.
2. На плоскости $ХОУ$ задана своими координатами точка A . Указать, где она расположена: на какой оси или в какой координатной четверти.
3. Грузовой автомобиль выехал из одного города в другой со скоростью v_1 км/ч. Через t ч в этом же направлении выехал легковой автомобиль со скоростью v_2 км/ч. Составить программу, определяющую, догонит ли легковой автомобиль грузовой через t_1 ч после своего выезда.
4. Написать программу нахождения суммы большего и меньшего из 3 чисел.



5. Написать программу, распознающую по длинам сторон среди всех треугольников прямоугольные. Если таковых нет, то вычислить величину угла C .
6. Найти $\max\{\min(a, b), \min(c, d)\}$.
7. Составить программу, осуществляющую перевод величин из радианной меры в градусную или наоборот. Программа должна запрашивать, какой перевод нужно осуществить, и выполнять указанное действие.

Уровень 2

8. Заданы размеры A, B прямоугольного отверстия и размеры x, y, z кирпича. Определить, пройдет ли кирпич через отверстие.
9. Два прямоугольника, расположенные в первом квадранте, со сторонами, параллельными осям координат, заданы координатами своих левого верхнего и правого нижнего углов. Для первого прямоугольника это точки (x_1, y_1) и $(x_2, 0)$, для второго — (x_3, y_3) , $(x_4, 0)$. Составить программу, определяющую, пересекаются ли данные прямоугольники, и вычисляющую площадь общей части, если они пересекаются.
10. В небоскребе N этажей и всего один подъезд; на каждом этаже по 3 квартиры; лифт может останавливаться только на нечетных этажах. Человек садится в лифт и набирает номер нужной ему квартиры M . На какой этаж должен доставить лифт пассажира?
11. Написать программу, которая по заданным трем числам определяет, является ли сумма каких-либо двух из них положительной.
12. Известно, что из четырех чисел a_1, a_2, a_3 и a_4 одно отлично от трех других, равных между собой; присвоить номер этого числа переменной n .
13. Составить программу, которая проверяла бы, не приводит ли суммирование двух целых чисел A и B к переполнению (т. е. к результату большему, чем 32 767). Если будет переполнение, то сообщить об этом, иначе вывести сумму этих чисел.

Уровень 3

14. Заданы координаты (на плоскости) вершин четырехугольника. Определить, является ли он: а) ромбом; б) параллелограммом; в) прямоугольником; г) квадратом.

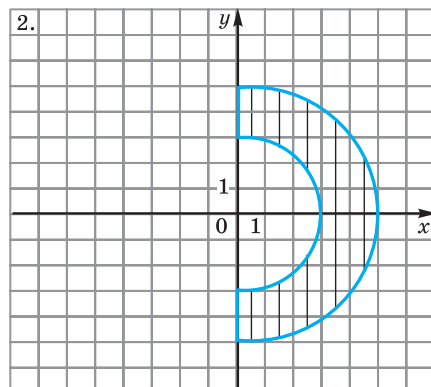
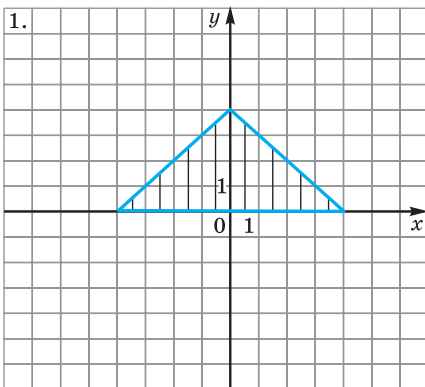
15. Для двух треугольных пирамид, заданных ребрами, определить, площадь полной поверхности которой из них больше и на сколько.
16. Дана точка $A(x, y)$. Определить, принадлежит ли она треугольнику с вершинами в точках (x_1, y_1) , (x_2, y_2) , (x_3, y_3) .
17. Написать программу, определяющую, будут ли прямые $A_1x + B_1y + C_1 = 0$ и $A_2x + B_2y + C_2 = 0$ перпендикулярны. Если нет, то найти угол между ними.
18. Заданы координаты вершин прямоугольника: (x_1, y_1) , (x_2, y_2) , (x_3, y_3) , (x_4, y_4) . Определить площадь части прямоугольника, расположенной в I координатной четверти.
19. Найти координаты точек пересечения прямой $y = kx + b$ и окружности радиуса R с центром в начале координат. В каких координатных четвертях находятся точки пересечения? Если точек пересечения нет или прямая касается окружности, выдать соответствующее сообщение.
20. Дана точка $A(x, y)$. Определить, принадлежит ли она прямоугольнику с вершинами в точках (x_1, y_1) , (x_2, y_2) , (x_3, y_3) , (x_4, y_4) .

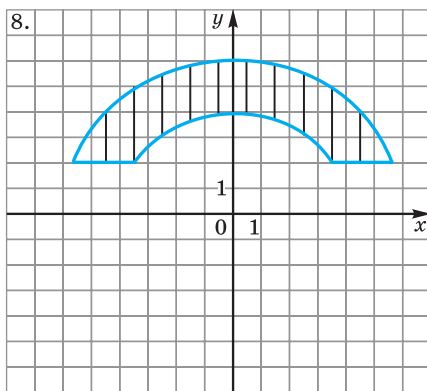
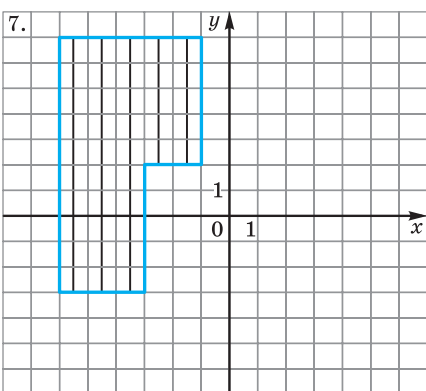
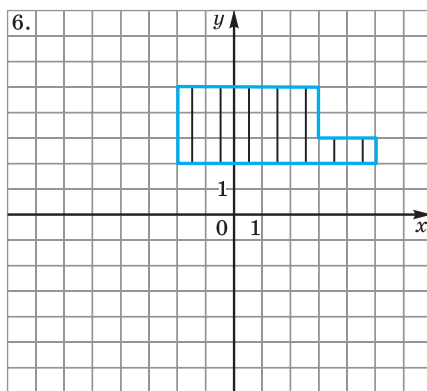
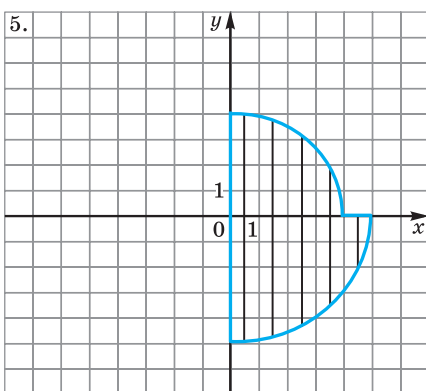
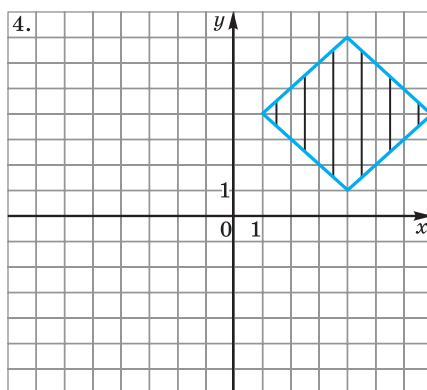
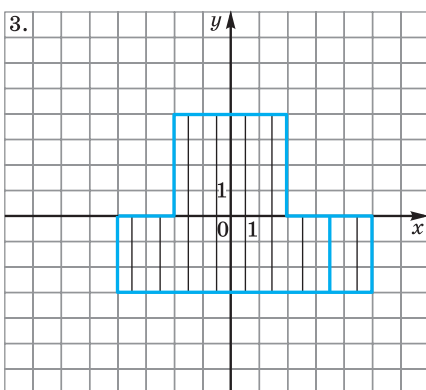
Задание 2

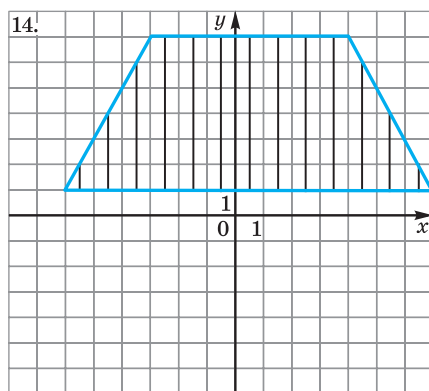
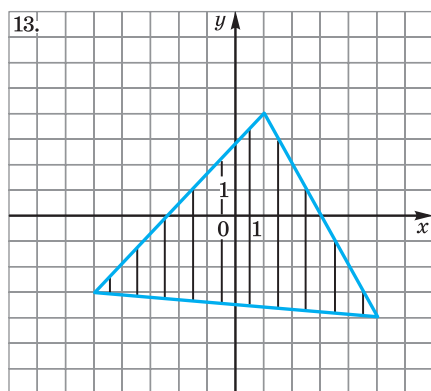
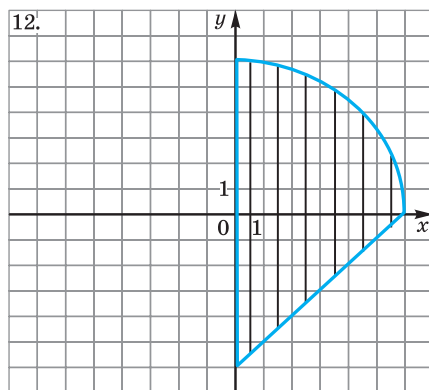
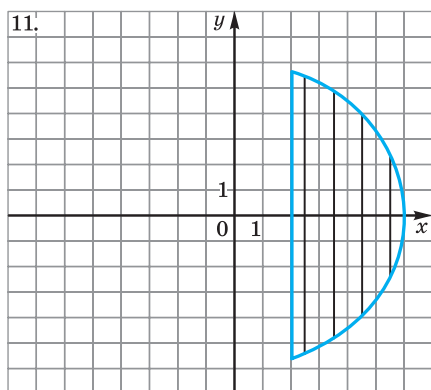
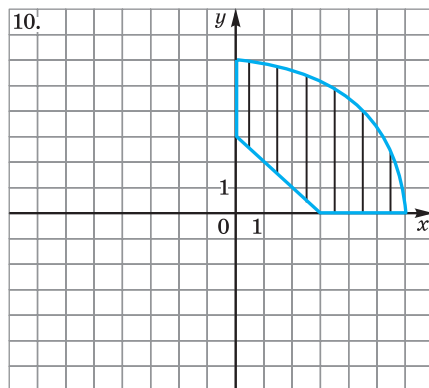
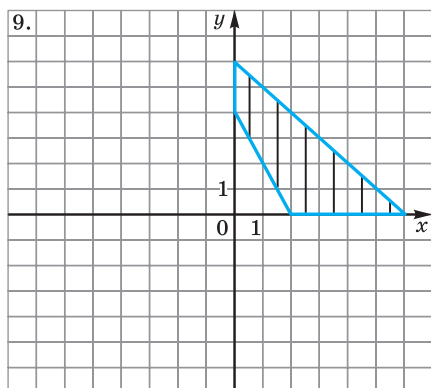
Задачи на определение принадлежности точки области

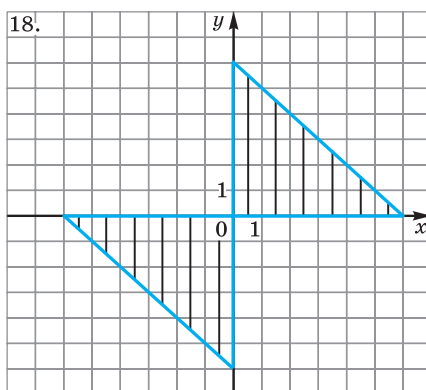
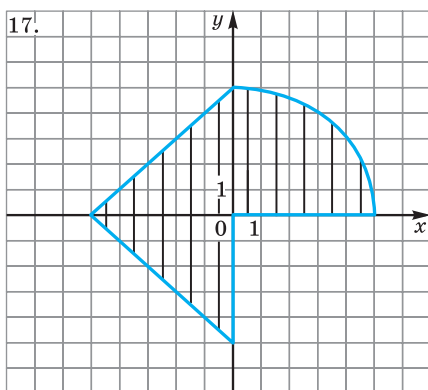
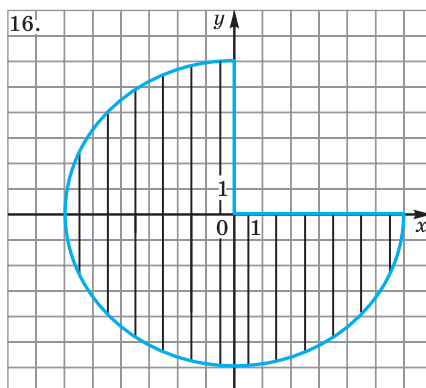
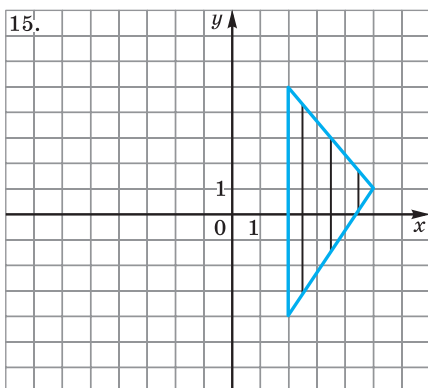
Для каждой задачи составить программу, содержащую ветвления и определяющую, принадлежит ли точка с координатами (X, Y) заштрихованной области.

Уровень 2









Задание 3

Задачи на использование оператора выбора

Для каждой задачи составить программу с ветвящейся структурой, используя оператор выбора `SELECT CASE`.

Уровень 1

1. Для каждой введенной цифры (0–9) вывести соответствующее ей название на английском языке (0 — zero, 1 — one, 2 — two, ...).
2. Составить программу, которая по данному числу (1–12) выводит название соответствующего ему месяца на английском языке.

3. Написать программу, которая по вводимому числу от 1 до 11 (номеру класса) выдает соответствующее сообщение «Привет, k -классник». Например, если $k = 1$, «Привет, первоклассник»; при $k = 4$: «Привет, четвероклассник».
4. Составить программу, позволяющую получить словесное описание школьных отметок (1 — плохо, 2 — неудовлетворительно, 3 — удовлетворительно, 4 — хорошо, 5 — отлично).
5. Написать программу, которая бы по введенному номеру единицы измерения (1 — дециметр, 2 — километр, 3 — метр, 4 — миллиметр, 5 — сантиметр) и длине отрезка L выдавала соответствующее значение длины отрезка в метрах.
6. Написать программу, которая бы по введенному номеру единицы измерения (1 — килограмм, 2 — миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер) и массе M выдавала соответствующее значение массы в килограммах.
7. Даны два действительных положительных числа x и y . Арифметические действия над числами пронумерованы (1 — сложение, 2 — вычитание, 3 — умножение, 4 — деление). Составить программу, которая по введенному номеру выполняет то или иное действие над числами.
8. Написать программу, которая по номеру дня недели (целому числу от 1 до 7) выдает в качестве результата расписание уроков в вашем классе в этот день.

Уровень 2

9. Составить программу, которая по заданному году и номеру месяца определяет количество дней в этом месяце.
10. Пусть элементами круга являются радиус (первый элемент), диаметр (второй элемент) и длина окружности (третий элемент). Составить программу, которая по номеру элемента запрашивала бы его соответствующее значение и вычисляла бы площадь круга.
11. Пусть элементами прямоугольного равнобедренного треугольника являются: 1) катет a ; 2) гипотенуза b ; 3) высота, опущенная из вершины прямого угла на гипотенузу h ; 4) площадь S . Составить программу, которая по заданному номеру и значению соответствующего элемента вычисляла бы значение всех остальных элементов треугольника.

12. В старояпонском календаре был принят 12-летний цикл. Годы внутри цикла носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. Написать программу, которая вводит номер некоторого года и печатает его название по старояпонскому календарю. (Справка: 1996 г. — год Крысы — начало очередного цикла.)
13. Для целого числа k от 1 до 99 напечатать фразу «Мне k лет», учитывая при этом, что при некоторых значениях k слово «лет» надо заменить на слово «год» или «года». Например, 11 лет, 22 года, 51 год.
14. Написать программу, которая по введенному числу от 1 до 12 (номеру месяца) выдает все приходящиеся на этот месяц праздничные дни (например, если введено число 1, то: 1 января — Новый год, 7 января — Рождество).
15. Дано натуральное число N . Если оно делится на 4, вывести на экран ответ $N = 4k$ (где k — соответствующее частное); если остаток от деления на 4 равен 1, $N = 4k + 1$; если остаток от деления на 4 равен 2, $N = 4k + 2$; если остаток от деления на 4 равен 3, $N = 4k + 3$. Например, $12 = 4 \cdot 3$, $22 = 4 \cdot 5 + 2$.
16. Пусть элементами равностороннего треугольника являются: 1) сторона a ; 2) площадь S ; 3) высота h ; 4) радиус вписанной окружности r ; 5) радиус описанной окружности R . Составить программу, которая по заданному номеру и значению соответствующего элемента вычисляла бы значение всех остальных элементов треугольника.

Работа 3.4. Программирование циклических алгоритмов

Задание 1

Циклы с заданным числом повторений

Вычислить значение суммы или произведения числовой последовательности.

Уровень 1

1. Дано натуральное число N . Вычислить:

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{N}.$$

2. Дано натуральное число N . Вычислить:

$$S = 1 + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \dots + \frac{1}{2N+1}.$$

3. Дано натуральное число N . Вычислить:

$$S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \dots + (-1)^N \frac{1}{2^N}.$$

4. Дано натуральное число N . Вычислить:

$$S = \frac{1}{\sin 1} + \frac{1}{\sin 1 + \sin 2} + \dots + \frac{1}{\sin 1 + \sin 2 + \dots + \sin N}.$$

5. Дано натуральное число N . Вычислить произведение первых N сомножителей:

$$P = \frac{2}{3} \cdot \frac{4}{5} \cdot \frac{6}{7} \cdot \dots \cdot \frac{2N}{2N+1}.$$

6. Дано натуральное n . Вычислить:

$$\frac{2}{1} + \frac{3}{2} + \frac{4}{3} + \dots + \frac{n+1}{n}.$$

7. Вычислить:

$$(1 + \sin 0,1)(1 + \sin 0,2) \cdot \dots \cdot (1 + \sin 10).$$

Уровень 2

8. Дано натуральное число N . Вычислить:

$$\frac{\cos 1}{\sin 1} \cdot \frac{\cos 1 + \cos 2}{\sin 1 + \sin 2} \cdot \dots \cdot \frac{\cos 1 + \cos 2 + \dots + \cos N}{\sin 1 + \sin 2 + \dots + \sin N}.$$

9. Дано действительное число x . Вычислить:

$$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \frac{x^{13}}{13!}.$$

10. Даны натуральное n , действительное x . Вычислить:

$$S = \sin x + \sin \sin x + \dots + \underbrace{\sin \sin \dots \sin x}_{n \text{ раз}}.$$

11. Дано действительное x . Вычислить:

$$\frac{(x-1)(x-3)(x-7) \cdot \dots \cdot (x-63)}{(x-2)(x-4)(x-8) \cdot \dots \cdot (x-64)}.$$

12. Дано натуральное n . Вычислить:

$$S = 1 \cdot 2 + 2 \cdot 3 \cdot 4 + \dots + n \cdot (n + 1) \cdot \dots \cdot 2n.$$

13. Дано натуральное число n . Вычислить:

$$P = \left(1 - \frac{1}{2^2}\right) \left(1 - \frac{1}{3^2}\right) \cdot \dots \cdot \left(1 - \frac{1}{n^2}\right), \quad \text{где } n > 2.$$

14. Вычислить по схеме Горнера:

$$y = x^{10} + 2x^9 + 3x^8 + \dots + 10x + 11.$$

15. Числа Фибоначчи (f_n) определяются формулами

$$f_0 = f_1 = 1, \quad f_n = f_{n-1} + f_{n-2} \quad \text{при } n = 2, 3, \dots$$

Для данного значения p определить f_p .

16. Даны натуральные числа n и k . Вычислить:

$$\sqrt{k + \sqrt{2k + \dots + \sqrt{k(n-1) + \sqrt{kn}}}}.$$

Задание 2

Итерационные циклы

Найти наименьший номер последовательности, заданной рекуррентной формулой, для которого выполняется условие $|a_n - a_{n-1}| < \varepsilon$, где ε — малая величина. Вывести на экран этот номер и все элементы a_i , где $i = 1, 2, \dots, n$.

Уровень 2

$$1. \quad a_n = \operatorname{arctg} a_{n-1} + 1, \quad a_1 = 0. \quad 5. \quad a_n = \frac{2 + a_{n-1}^2}{2a_{n-1}}, \quad a_1 = 2.$$

$$2. \quad a_n = 2 + \frac{1}{a_{n-1}}, \quad a_1 = 2. \quad 6. \quad a_n = \frac{a_{n-1} + a_{n-2}}{2}, \quad a_1 = 1, \quad a_2 = 2.$$

$$3. \quad a_n = \frac{1}{2} \operatorname{tg} a_{n-1}, \quad a_1 = 0,5. \quad 7. \quad a_n = e^{-a_{n-1}}, \quad a_1 = 0.$$

$$4. \quad a_n = \frac{1}{2} \cos a_{n-1}, \quad a_1 = 0,5. \quad 8. \quad a_n = \frac{x}{2a_{n-1}^2}, \quad a_1 = x.$$

Задание 3

Циклы при обработке целых чисел

Решить поставленные задачи с помощью циклических алгоритмов (простых и вложенных), используя операции целочисленной арифметики.

Уровень 2

1. Натуральные числа a , b , c называются числами Пифагора, если выполняется условие $a^2 + b^2 = c^2$. Напечатать все числа Пифагора, меньшие N .
2. Найти наибольшую и наименьшую цифры в записи данного натурального числа.
3. Дано натуральное число N . Найти и вывести все числа в интервале от 1 до $N - 1$, у которых сумма всех цифр совпадает с суммой цифр данного числа. Если таких чисел нет, то вывести слово «нет».

Пример: $N = 44$. Числа: 17, 26, 35.

4. Дано натуральное число N . Найти и вывести все числа в интервале от 1 до $N - 1$, у которых произведение всех цифр совпадает с суммой цифр данного числа. Если таких чисел нет, то вывести слово «нет».

Пример: $N = 44$. Числа: 18, 24.

5. Дано натуральное число N ($N > 9$). Определить количество нулей, идущих подряд в младших разрядах данного числа.

Пример: $N = 1020000$. Количество нулей равно четырем.

6. Найти все натуральные числа, не превосходящие заданного n , которые делятся на каждую из своих цифр.
7. Дано натуральное число N . Получить новое число M , которое образуется из числа N путем замены последней цифры на наименьшую цифру в записи числа N .

Пример: $N = 128452$, $M = 128451$.

8. Дано натуральное число N . Получить новое число M , которое образуется из числа N путем замены последней цифры на наибольшую цифру в записи числа N .

Пример: $N = 128452$, $M = 128458$.

Уровень 3

9. Даны два натуральных числа m и n . Проверить, есть ли в записи числа m цифры, одинаковые с цифрами в записи числа n .
10. Дано натуральное число N ($N > 9$). Определить количество нулей в цифровой записи числа, кроме нулей в младших разрядах.

Пример: $N = 10025000$. Количество нулей равно двум.

11. Натуральное число M называется совершенным, если оно равно сумме всех своих делителей, включая 1, но исключая себя. Напечатать все совершенные числа, меньшие заданного числа N .
12. Дано целое $n > 2$. Напечатать все простые числа из диапазона $[2, n]$.

Работа 3.5. Программирование с использованием подпрограмм

Задание 1

Для решения всех задач сделать два варианта программы: с реализацией указанной подпрограммы в виде функции и в виде процедуры.

Уровень 1

1. Составить программу нахождения наибольшего общего делителя (НОД) и наименьшего общего кратного (НОК) двух натуральных чисел $\text{НОК}(A, B) = \frac{A \cdot B}{\text{НОД}(A, B)}$. Использовать подпрограмму алгоритма Евклида для определения НОД.
2. Вычислить площадь правильного шестиугольника со стороной a , используя подпрограмму вычисления площади треугольника.
3. Даны две дроби $\frac{A}{B}$ и $\frac{C}{D}$ (A, B, C, D — натуральные числа). Составить программу деления дроби на дробь. Ответ должен быть несократимой дробью. Использовать подпрограмму алгоритма Евклида для определения НОД.

4. Даны две дроби $\frac{A}{B}$ и $\frac{C}{D}$ (A, B, C, D — натуральные числа). Составить программу умножения дроби на дробь. Ответ должен быть несократимой дробью. Использовать подпрограмму алгоритма Евклида для определения НОД.
5. Даны две дроби $\frac{A}{B}$ и $\frac{C}{D}$ (A, B, C, D — натуральные числа). Составить программу вычитания из первой дроби второй. Ответ должен быть несократимой дробью. Использовать подпрограмму алгоритма Евклида для определения НОД.
6. Написать программу вычисления суммы $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ для заданного числа n . Результат представить в виде несократимой дроби $\frac{p}{q}$ (p, q — натуральные). Использовать подпрограммы алгоритма Евклида для определения НОД и сложения двух простых дробей.
7. Даны числа X, Y, Z, T — длины сторон четырехугольника. Вычислить его площадь, если угол между сторонами длиной X и Y — прямой. Использовать две подпрограммы для вычисления площадей: прямоугольного треугольника и прямоугольника.

Задание 2

Для всех задач выделить подзадачи, решения которых могут быть реализованы через подпрограммы. Выбрать наиболее удобный вариант подпрограммы: функцию или процедуру. Составить программу решения задачи.

Уровень 2

1. Дано простое число. Найти следующее за ним простое число.
2. Для заданного натурального числа n найти наименьший нечетный натуральный делитель k ($k \neq 1$).
3. Заменить данное натуральное число на число, которое получается из исходного записью его цифр в обратном порядке (например, дано число 156, нужно получить 651).
4. Найти все натуральные числа, не превосходящие заданного n , которые делятся на каждую из своих цифр.
5. Имеется часть катушки с автобусными билетами. Номер билета шестизначный. Составить программу, определяющую количество счастливых билетов на катушке, если меньший номер билета — N ,

больший — M (билет является счастливым, если сумма первых трех его цифр равна сумме последних трех).

6. Из заданного числа вычли сумму его цифр. Из результата вновь вычли сумму его цифр и т. д. Через сколько таких действий получится ноль?
7. На отрезке $[100, N]$ ($2^{10} < N < 2^{31}$) найти количество чисел, составленных из цифр a, b, c .
8. Найти все натуральные n -значные числа, цифры в которых образуют строго возрастающую последовательность (например, 1234, 5789).

Уровень 3

9. Два простых числа называются «близнецами», если они отличаются друг от друга на 2 (например, 41 и 43). Напечатать все пары «близнецов» из отрезка $[n, 2n]$, где n — заданное натуральное число, большее 2.
10. Дано четное число $n > 2$. Проверить для него гипотезу Гольдбаха: каждое четное n представляется в виде суммы двух простых чисел.
11. Составить программу разложения данного натурального числа на простые множители. Например, $200 = 2^3 \cdot 5^2$.
12. Дано натуральное число n . Найти все меньшие n числа Мерсенна. (Простое число называется числом Мерсенна, если оно может быть представлено в виде $2^p - 1$, где p — тоже простое число. Например, $31 = 2^5 - 1$ — число Мерсенна.)

Работа 3.6. Программирование обработки одномерных массивов

Задание

Составить программу решения поставленной задачи по обработке одномерного массива (вектора). По возможности, использовать подпрограммы.

Уровень 1

1. Дана последовательность действительных чисел $a_1, a_2, \dots, a_n$. Выяснить, будет ли она возрастающей.
2. Дан массив из N действительных чисел. Подсчитать, сколько в нем отрицательных, положительных и нулевых элементов.

3. Даны действительные числа $a_1, a_2, \dots, a_n$. Поменять местами первый наибольший элемент с последним наименьшим элементом.
4. В заданном одномерном массиве поменять местами соседние элементы, стоящие на четных местах, с элементами, стоящими на нечетных местах.
5. Задана последовательность $\{X_i\}$ из N вещественных чисел. Вычислить последовательность $\{S_i\}$ по формуле:

$$S_i = \sqrt{\frac{(X_i - M)^2}{N - 1}},$$

где M — среднее арифметическое значение последовательности X .

6. Задана последовательность из N целых чисел. Вычислить сумму тех элементов массива, порядковые номера которых совпадают со значением этого элемента.
7. Определить, сколько процентов от всего количества элементов последовательности целых чисел составляют нечетные элементы.
8. Дан массив $X[N]$ целых чисел. Не используя других массивов, переставить его элементы в обратном порядке.

Уровень 2

9. Задана последовательность из N вещественных чисел. Вычислить сумму чисел, порядковые номера которых являются простыми числами.
10. Последовательность $a_1, a_2, \dots, a_n$ состоит из нулей и единиц. Поместить в начало этой последовательности все нули, а затем все единицы.
11. Даны действительные числа $a_1, a_2, \dots, a_{2n}$. Найти:

$$\max(a_1 + a_{2n}, a_2 + a_{2n-1}, \dots, a_n + a_{n+1}).$$

12. Дана последовательность действительных чисел $a_1 \leq a_2 \leq \dots \leq a_n$. Вставить действительное число b в нее так, чтобы последовательность осталась неубывающей.
13. Дана последовательность целых чисел $a_1, a_2, \dots, a_n$. Указать пары чисел a_i, a_j , таких что $a_i + a_j = m$, где m — заданное целое число.
14. Даны координаты n ($n \leq 30$) точек на плоскости: $(X_1, Y_1), \dots, (X_n, Y_n)$. Найти номера пары точек, расстояние между которыми наибольшее (считать, что такая пара единственная).

15. Дан массив, состоящий из n натуральных чисел. Образовать новый массив, элементами которого будут элементы исходного, оканчивающиеся на цифру k .
16. Дан массив целых чисел. Найти в этом массиве минимальный элемент m и максимальный элемент M . Получить в порядке возрастания все целые числа из интервала $(m; M)$, которые не входят в данный массив.
17. Даны две последовательности $a_1, a_2, \dots, a_n$ и $b_1, b_2, \dots, b_m$ ($m < n$). В каждой из них значения элементов различны. Верно ли, что все элементы второй последовательности входят в первую последовательность?
18. Вывести значения и номера наибольшего, наименьшего и наименее удаленного от среднего арифметического значения элементов данной последовательности вещественных чисел.

Уровень 3

19. Сформировать массив простых чисел, не больших заданного натурального числа N .
20. Сформировать массив простых множителей заданного числа.
21. В одномерном массиве все отрицательные элементы переместить в начало массива, а остальные — в конец с сохранением порядка следования. Дополнительный массив заводить не разрешается.
22. В одномерном массиве с четным количеством элементов ($2N$) находятся координаты N точек плоскости. Они располагаются в следующем порядке: $x_1, y_1, x_2, y_2, x_3, y_3$, и т. д. Определить:
 - а) минимальный радиус окружности с центром в начале координат, которая содержит все точки;
 - б) внутренний и внешний радиусы кольца с центром в начале координат, которое содержит все точки;
 - в) номера точек, которые могут являться вершинами квадрата;
 - г) номера точек, которые могут являться вершинами равнобедренного треугольника;
 - д) номера самых удаленных и наименее удаленных друг от друга точек;
 - е) три точки, которые являются вершинами треугольника, для которого разность точек вне его и внутри является минимальной.
23. Дана последовательность целых чисел. Найти количество различных чисел в этой последовательности.

24. На плоскости n точек заданы своими координатами, и также дана окружность радиуса R с центром в начале координат. Указать множество всех треугольников с вершинами в заданных точках, пересекающихся с окружностью; множество всех треугольников, содержащихся внутри окружности.
25. Разделить массив на две части, поместив в первую элементы, большие среднего арифметического элементов массива, а во вторую — меньшие (части не сортировать).
26. Даны две последовательности $a_1 \leq a_2 \leq \dots \leq a_n$ и $b_1 \leq b_2 \leq \dots \leq b_m$. Образовать из них новую последовательность чисел так, чтобы она тоже была неубывающей.

Примечание. Дополнительный массив не использовать.

27. *Сортировка вставками.* Дана последовательность чисел $a_1, a_2, \dots, a_n$. Требуется переставить числа в порядке возрастания. Делается это следующим образом. Пусть $a_1, a_2, \dots, a_i$ — упорядоченная по неубыванию последовательность, т. е. $a_1 \leq a_2 \leq \dots \leq a_i$. Берется следующее число a_{i+1} и вставляется в последовательность так, чтобы новая последовательность была также возрастающей. Процесс производится до тех пор, пока все элементы от $i + 1$ до n не будут перебраны.

Примечание. Место помещения очередного элемента в отсортированную часть найти с помощью двоичного поиска. Двоичный поиск оформить в виде отдельной функции.

28. *Алгоритм сортировки фон Неймана.* Упорядочить массив $a_1, a_2, \dots, a_n$ по неубыванию с помощью алгоритма сортировки слияниями:

- 1) каждая пара соседних элементов сливается в одну группу из двух элементов (последняя группа может состоять из одного элемента);
- 2) каждая пара соседних двухэлементных групп сливается в одну четырехэлементную группу и т. д.

При каждом слиянии новая укрупненная группа упорядочивается.

29. *Шейкер-сортировка.* Алгоритм «пузырьковой» сортировки легко улучшить. Разумно запомнить, производился ли на данном проходе какой-либо обмен. Если нет, то алгоритм можно закончить. Еще одно улучшение заключается в том, что периодически меняется направление сортировки. Это позволяет бороться с некоторой асимметрией «пузырькового» метода. Написать программу, реализующую данный улучшенный алгоритм.

Работа 3.7. Программирование обработки двумерных массивов

Задание

Составить программу решения поставленной задачи по обработке двумерного массива (матрицы). По возможности, использовать подпрограммы.

Уровень 1

1. Вычислить сумму и число положительных элементов матрицы $A[N, N]$, находящихся над главной диагональю.
2. Дана целая квадратная матрица n -го порядка. Определить, является ли она магическим квадратом, т. е. такой матрицей, в которой суммы элементов во всех строках и столбцах одинаковы.
3. Определить, является ли заданная целая квадратная матрица n -го порядка симметричной (относительно главной диагонали).
4. Дана целочисленная квадратная матрица. Найти в каждой строке наибольший элемент и поменять его местами с элементом главной диагонали в этой же строке.
5. Упорядочить по возрастанию элементы каждой строки матрицы размером $n \times m$.
6. Задана квадратная матрица. Получить транспонированную матрицу (перевернутую относительно главной диагонали).
7. Квадратная матрица, симметричная относительно главной диагонали, задана верхним треугольником в виде одномерного массива. Восстановить исходную матрицу и напечатать по строкам.
8. Задана матрица порядка n и число k . Разделить элементы k -й строки на диагональный элемент, расположенный в этой строке.
9. Для целочисленной квадратной матрицы найти число элементов, кратных k , и наибольший из этих элементов.
10. Найти наибольший и наименьший элементы прямоугольной матрицы и поменять их местами.
11. В данной действительной квадратной матрице порядка n найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.

12. Дана действительная матрица размером $n \times m$. Требуется преобразовать матрицу: поэлементно вычесть последнюю строку из всех строк, кроме последней.
13. Определить наименьший элемент каждой четной строки матрицы $A[M, N]$.

Уровень 2

14. Задана квадратная матрица. Переставить строку с максимальным элементом на главной диагонали со строкой с заданным номером m .
15. Определить номера строк матрицы $R[M, N]$, хотя бы один элемент которых равен c , и элементы этих строк умножить на d .
16. Дана матрица $B[N, M]$. Найти в каждой строке матрицы максимальный и минимальный элементы и поменять их с первым и последним элементами строки соответственно.
17. Элемент матрицы назовем *седловой точкой*, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или, наоборот, является наибольшим в своей строке и наименьшим в своем столбце. Для заданной целой матрицы размером $n \times m$ напечатать индексы всех ее седловых точек.
18. Дана вещественная матрица размером $n \times m$. Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент (или один из них) оказался в верхнем левом углу.
19. Дана квадратная матрица $A[N, N]$. Записать на место отрицательных элементов матрицы нули, а на место положительных — единицы. Вывести на печать нижнюю треугольную матрицу в общепринятом виде.
20. Дана действительная матрица размером $n \times m$, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.
21. Дана действительная квадратная матрица порядка N (N — нечетное), все элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.

22. Для заданной квадратной матрицы сформировать одномерный массив из ее диагональных элементов. Найти след матрицы, просуммировав элементы одномерного массива. Преобразовать исходную матрицу по правилу: четные строки разделить на полученное значение, нечетные оставить без изменения.
23. Дана прямоугольная матрица. Найти строку с наибольшей и строку с наименьшей суммой элементов. Вывести на печать найденные строки и суммы их элементов.
24. В данной действительной квадратной матрице порядка n найти наибольший по модулю элемент. Получить квадратную матрицу порядка $n - 1$ путем отбрасывания из исходной матрицы строки и столбца, на пересечении которых расположен элемент с найденным значением.
25. Расположить столбцы матрицы $D[M, N]$ в порядке возрастания элементов k -й строки ($1 \leq k \leq M$).

Уровень 3

26. Среди столбцов заданной целочисленной матрицы, содержащих только такие элементы, которые по модулю не больше 10, найти столбец с минимальным произведением элементов.
27. Для заданной квадратной матрицы найти такие k , что k -я строка матрицы совпадает с k -м столбцом.
28. Найти максимальный среди всех элементов тех строк заданной матрицы, которые упорядочены (либо по возрастанию, либо по убыванию).
29. Составить программу, которая заполняет квадратную матрицу порядка n натуральными числами 1, 2, 3, ..., n^2 , записывая их в нее «по спирали».

Например, для $n = 5$ получаем следующую матрицу:

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

30. Среди тех строк целочисленной матрицы, которые содержат только нечетные элементы, найти строку с максимальной суммой модулей элементов.
31. Подсчитать количество строк заданной целочисленной матрицы $N \times N$, являющихся перестановкой чисел 1, 2, ..., N (т. е. содержащих каждое из чисел 1, 2, ..., N ровно один раз).

Работа 3.8. Программирование обработки строк символов

Задание

Составить на Паскале программу решения поставленной задачи по обработке символьных строк. По возможности, использовать подпрограммы. В последующих задачах подразумевается, что слова в тексте (в строке) отделяются друг от друга пробелами.

Уровень 1

1. Дана строка, заканчивающаяся точкой. Подсчитать, сколько слов в строке.
2. Дана строка, содержащая английский текст. Найти количество слов, начинающихся с буквы «b».
3. В строке заменить все двоеточия (:) точкой с запятой (;). Подсчитать количество замен.
4. Дана строка. Преобразовать ее, заменив звездочками все двоеточия (:), встречающиеся среди первых $n/2$ символов, и заменив точками все восклицательные знаки, встречающиеся среди символов, стоящих после $n/2$ символов. Здесь n — длина строки.
5. В строке удалить символ двоеточие (:) и подсчитать количество удаленных символов.
6. Дана строка символов, среди которых есть одна открывающаяся и одна закрывающаяся скобки. Вывести на экран все символы, расположенные внутри этих скобок.
7. Дана строка, содержащая текст. Найти длину самого короткого и самого длинного слов.
8. Дана строка, содержащая текст, заканчивающийся точкой. Вывести на экран все слова, содержащие три буквы. Если таких слов нет, то вывести сообщение об этом.
9. Дана строка. Преобразовать ее, удалив каждый символ * и повторив дважды подряд каждый символ, отличный от *.
10. Дана строка текста. Подсчитать количество букв «k» в последнем слове.
11. Определить, сколько раз в тексте встречается заданное слово.

Уровень 2

12. Дана строка-предложение на английском языке. Преобразовать строку так, чтобы каждое слово начиналось с заглавной буквы.
13. Дана строка. Подсчитать, сколько различных символов встречается в ней. Вывести их на экран.
14. Дана строка. Подсчитать самую длинную последовательность подряд идущих букв «а».
15. Имеется строка, содержащая буквы латинского алфавита и цифры. Вывести на экран длину наибольшей последовательности цифр, идущих подряд.
16. Дана строка. Указать те слова, которые содержат хотя бы одну букву «k».
17. Дана строка. Найти в ней те слова, которые начинаются и оканчиваются одной и той же буквой.
18. Строка содержит одно слово. Проверить, будет ли оно читаться одинаково справа налево и слева направо (т. е. является ли оно палиндромом).
19. В записке слова зашифрованы — каждое из них записано наоборот. Расшифровать сообщение.
20. *Статистика*. Дан текст. Написать программу, определяющую процентное отношение строчных и прописных букв к общему числу символов в нем.
21. Проверить, одинаковое ли число открывающихся и закрывающихся скобок в данной строке и правильно ли они расставлены, т. е. для каждой открывающей скобки есть соответствующая закрывающая скобка.

Уровень 3

22. Из заданной символьной строки выбрать те символы, которые встречаются в ней только один раз, в том порядке, в котором они встречаются в тексте.
23. *Расстояние между двумя словами равной длины* — это количество позиций, в которых эти слова различаются. В заданном предложении найти пару слов заданной длины с максимальным расстоянием.
24. *Форматирование текста*. Дан текст, состоящий из предложений, разделяемых точками. Напишите программу, производящую следующее форматирование: после каждой точки в конце предложения должен стоять хотя бы один пробел; первое слово в предложении должно начинаться с прописной буквы.
Замечание. Текст может быть как на русском, так и на английском языке.

25. *Лишние пробелы.* Дана строка, состоящая из слов, разделенных пробелами. Напишите программу, удаляющую лишние пробелы. Пробел считается лишним, если он:
- стоит в начале строки;
 - стоит в конце строки;
 - следует за пробелом.
26. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова (разделенные одним пробелом), но расположенные в обратном порядке.
27. Составить программу преобразования натуральных чисел, записанных в римской нумерации, в десятичную систему счисления.
28. а) Дана строка-предложение. Зашифровать ее, поместив вначале все символы, расположенные на четных местах, а затем, в обратном порядке, все символы, расположенные на нечетных местах. (Например, строка 'Программа' превратится в 'ргамамроП'.)
- б) Запрограммировать решение обратной задачи (расшифровки).

Работа 3.9. Программирование обработки записей

Задание

Исходя из условия задачи, определить структуру комбинированного типа данных и типы полей. Исходные данные разместить в текстовом файле. В программе не использовать массивов записей. Результаты выводить на экран и в текстовый файл.

Уровень 1

1. Из данного списка спортсменов получить данные о тех из них, кто занимается плаванием. Указать возраст, сколько лет они занимаются спортом.
2. Из ассортимента конфет, выпускаемых кондитерской фабрикой, выбрать те, стоимость которых от 100 до 200 руб. за 1 кг. Указать срок их годности и номера магазинов, в которых они имеются в продаже.
3. Получить список учеников музыкальной школы, которые учатся играть на скрипке. Указать также, сколько лет они занимаются музыкой и принимали ли участие в каких-либо конкурсах.

4. Получить фамилии детей данного детского сада, которые родились в указанном месяце; вывести также их возраст и группу.
5. Получить список тех учителей школы, которые преподают математику и информатику, указать стаж их работы и недельную нагрузку.
6. Получить анкетные данные учеников, участвовавших в олимпиаде по информатике и заработавших не менее 30 баллов.
7. Даны результаты переписи населения. Получить фамилии, имена и подсчитать общее число жителей, родившихся после 1990 года.
8. В сведениях об экспортируемых товарах указывается наименование товара, страна, импортирующая товар, и объем поставляемой партии в штуках. Составить список стран, в которые экспортируется данный товар, и общий объем его экспорта.
9. Сведения о сотрудниках включают: фамилию, имя, отчество, дату рождения, полученное образование, домашний адрес, профессию. Получить ФИО (фамилию, имя, отчество) всех сотрудников с высшим образованием. Выдать сведения о сотрудниках, имеющих данную профессию и родившихся не раньше указанной даты.
10. При поступлении в университет лица, получившие оценку «неудовлетворительно» на первом экзамене, ко второму экзамену не допускаются. Считая фамилии абитуриентов и их оценки после первого экзамена исходными данными, составить список абитуриентов, допущенных ко второму экзамену.

Уровень 2

11. Имеются сведения о датах рождения сотрудников учреждения.
 - а) Определить самого молодого сотрудника.
 - б) Определить самого старшего сотрудника.
 - в) Получить список всех сотрудников, родившихся весной.
12. Вычислить средний балл учеников класса, если известны оценки каждого ученика по математике, русскому языку и физике. Получить список учеников, имеющих средний балл выше среднего в классе.

13. Среди работников данного предприятия найти тех, чья заработная плата за месяц ниже средней по предприятию, а также распечатать список тех, кто проработал на предприятии более 10 лет, с указанием их фамилии, зарплаты, стажа работы и должности.
14. Получить фамилии тех учеников класса, которые являются хорошистами и отличниками по итогам года. Также указать, насколько их средний балл отличается от среднего балла класса.
15. Имеются сведения об учениках класса. Определить средний вес мальчиков и средний рост девочек. Кто из учеников класса самый высокий?
16. Составить программу назначения стипендии студентам по результатам сессии, используя следующие правила:
 - 1) если все оценки 5, назначается повышенная стипендия;
 - 2) если оценки 4 и 5, назначается обычная стипендия;
 - 3) если есть оценка 3, стипендия не назначается.
 В результате работы программы получить список группы с оценками и средним баллом каждого студента и два списка фамилий (назначенных на повышенную и обычную стипендию).
17. В таблице хранятся следующие данные об учениках: фамилия, имя, отчество, рост, вес. Вычислить средний рост учеников, найти самого высокого и самого низкого учеников. Сколько учеников могут заниматься в баскетбольной секции, если рост баскетболиста должен быть больше 170 см?
18. В столовой предлагаются N комплексных обедов, состоящих из Q блюд. Известна стоимость и калорийность каждого блюда. Сколько стоит самый дешевый и самый дорогой обед? Сколько калорий включает в себя самое калорийное блюдо?
19. N спортсменов-многоборцев принимают участие в соревнованиях по M видам спорта. По каждому виду спорта спортсмен набирает определенное количество очков. Вычислить, сколько очков в сумме набрал каждый спортсмен после окончания соревнований. Вычислить разницу в очках для спортсменов, занявших первое и последнее места.
20. N учеников проходили тестирование по M темам какого-либо предмета. Сколько очков набрал каждый ученик по всем темам? Вычислить средний балл, полученный учениками, и разницу между лучшим результатом и средним баллом.

21. Сведения о результатах сессии содержат следующую информацию: название предмета, номер группы, номер зачетной книжки, фамилия, имя, отчество студента, его оценки по итогам экзаменов. Получить отдельные списки: отличников, хорошистов, троечников и двоечников.

Уровень 3

22. На аптечном складе хранятся лекарства. Сведения о лекарствах содержатся в специальной ведомости: наименование лекарственного препарата; количество; цена; срок хранения (в месяцах). Выяснить, сколько стоит самый дорогой и самый дешевый препарат; сколько препаратов хранится на складе; какие препараты имеют срок хранения более 3 месяцев; сколько стоят все препараты, хранящиеся на складе.
23. Торговый склад производит уценку хранящейся продукции. Если продукция хранится на складе дольше n месяцев, то она уценивается в 2 раза, а если срок хранения превысил m ($m < n$) месяцев, но не достиг n , то в 1,5 раза. Получить ведомость уценки товаров, которая должна содержать следующую информацию: наименование товара, количество товара, цена товара до уценки, срок хранения товара, цена товара после уценки, общая стоимость товара до уценки, общая стоимость товара после уценки. Выяснить максимальный и минимальный сроки хранения товаров на складе; максимальную и минимальную цену товаров до уценки и после уценки.
24. Багаж пассажира характеризуется количеством вещей и общим весом вещей. В таблице представлены сведения о багаже нескольких пассажиров. Сведения о багаже каждого пассажира представляют собой запись с тремя полями: фамилия пассажира, количество вещей и вес всего багажа в килограммах.
- а) Найти пассажира, багаж которого имеет средний вес одной вещи, отличающийся не более чем на m кг от общего среднего веса одной вещи (для всех пассажиров).
 - б) Найти число пассажиров, имеющих более двух вещей, и число пассажиров, количество вещей которых превосходит среднее число вещей на одного пассажира.
 - в) Выяснить, имеется ли пассажир, багаж которого состоит из одной вещи весом менее m кг. Если такой пассажир есть, то вывести его фамилию и вес багажа.

25. Имеются сведения о книгах, содержащихся в библиотеке. Сведения о каждой из книг — это фамилия автора, название и год издания.
- а) Получить название книг данного автора, изданных с 1960 г.
 - б) Определить, имеется ли книга, содержащая в своем названии слово «информатика». Вывести сведения обо всех таких книгах.
 - в) Вывести сведения о книгах, которые имеют несколько изданий (автор и названия повторяются, а годы издания разные).
26. Сведения об игрушках, хранящихся на складе детского магазина, следующие: название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет).
- а) Получить названия игрушек, цена которых не превышает 500 руб. и которые подходят детям 5 лет.
 - б) Определить стоимость самого дорогого конструктора.
 - в) Получить список наиболее дорогих игрушек (цена которых отличается от цены самой дорогой игрушки не более чем на 200 руб.).
 - г) Получить названия игрушек, которые подходят детям как четырех, так и десяти лет.
 - д) Получить сведения о том, можно ли подобрать игрушку, любую, кроме мяча, подходящую ребенку трех лет.
 - е) Получить название самой дешевой игрушки.
 - ж) Получить название самой дорогой игрушки для детей до четырех лет.
 - з) Получить названия игрушек для детей четырех-пяти лет.
 - и) Получить название самой дешевой игрушки, подходящей детям двух-трех лет.
 - к) Определить стоимость самой дорогой куклы.
 - л) Определить стоимости всех кукол для детей шести лет.
 - м) Для детей какого возраста предназначается конструктор?
 - н) Для детей какого возраста предназначены кубики? Указать их среднюю стоимость.
27. Имеются сведения о программах телепередач на неделю: день недели, время, канал, вид и название телепередачи.
- а) Получить названия телепередач, которые идут в указанный день в указанный промежуток времени.
 - б) Получить названия телепередач, которые идут в указанный день на указанном канале.

- в) Получить информацию об указанном фильме. Если он отсутствует в телепрограмме, то вывести на экран сообщение «Такой передачи на данной неделе нет».
 - г) На каком канале и в какое время будет транслироваться развлекательная передача «Поле чудес»?
 - д) Выяснить, есть ли передача, транслирующаяся больше одного раза в одно и то же время. Если есть, то какая?
 - е) Получить названия телепередач, транслирующихся в указанное время на разных каналах.
 - ж) Получить название самой продолжительной передачи, которая идет в понедельник.
 - з) Получить информацию на каждый день недели о передаче, которая завершает эфир.
28. Имеются сведения о людях, которые ищут работу. Указываются специальность, опыт работы, образование, пол, возраст.
- а) Подобрать кандидатов на должность врача с опытом работы не менее пяти лет.
 - б) Найти работников с высшим экономическим образованием не старше 35 лет.
 - в) Найти работников, имеющих опыт работы в сфере торговли.
 - г) Получить полную информацию обо всех женщинах, возраст которых от 20 до 40 лет.
 - д) Определить средний возраст всех мужчин, ищущих работу.
 - е) Выяснить, кого в базе данных больше с высшим образованием — женщин или мужчин.
 - ж) Найти n самых молодых работников.

Оглавление

Введение	3
Глава 1. Информация	9
§ 1. Понятие информации	9
§ 2. Представление информации, языки, кодирование	13
§ 3. Измерение информации. Алфавитный подход	19
§ 4. Измерение информации. Содержательный подход	24
§ 5. Представление чисел в компьютере	32
§ 6. Представление текста, изображения и звука в компьютере	41
Глава 2. Информационные процессы	51
§ 7. Хранение информации	51
§ 8. Передача информации	57
§ 9. Обработка информации и алгоритмы	62
§ 10. Автоматическая обработка информации	67
§ 11. Информационные процессы в компьютере	72
Глава 3. Программирование обработки информации	84
§ 12. Алгоритмы и величины	84
§ 13. Структура алгоритмов	90
§ 14. Паскаль — язык структурного программирования	97
§ 15. Элементы языка Паскаль и типы данных	103
§ 16. Операции, функции, выражения	108
§ 17. Оператор присваивания, ввод и вывод данных	114
§ 18. Логические величины, операции, выражения	121
§ 19. Программирование ветвлений	130
§ 20. Пример поэтапной разработки программы решения задачи	134
§ 21. Программирование циклов	140
§ 22. Вложенные и итерационные циклы	148
§ 23. Вспомогательные алгоритмы и подпрограммы	153
§ 24. Массивы	161
§ 25. Организация ввода и вывода данных с использованием файлов	167
§ 26. Типовые задачи обработки массивов	173

§ 27. Символьный тип данных.	179
§ 28. Строки символов	183
§ 29. Комбинированный тип данных	188
Ответы на вопросы и задания	195
Практикум	207
Практические работы к главе 1 «Информация»	207
Практические работы к главе 2 «Информационные процессы»	222
Практические работы к главе 3 «Программирование обработки информации»	232